tanulmányok 113/1980

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

OPERÁCIÓS RENDSZEREK ELMÉLETE
VI. VISEGRÁDI TÉLI ISKOLA

Tanulmányok 113/1980.

MTA Számítástechnikai és Automatizálási Kutató Intézete
MTA Számítástudományi Bizottsága

ISBN 963 311 110 2
ISSN 0324-2951

Készült a KSH SZÁMOK
nyomdájában

237/1980.

A konferenciát a "Számítástechnika tudományos kérdései" c.
többoldalú akadémia együttmüködés keretében rendeztük.

Конференция была проведена в рамках
многостороннего сотрудничества академий
социалистических стран по проблеме
"Научные вопросы вычислительной техники"

Conference was held in the frame of the multilateral cooperation
of the academies of sciences of the socialist countries on Computer Sciences.

OPERATING SYSTEMS

The *Computer and Automation Institute* and the *Computer Science Committe of the Hungarian Academy of Sciences* have been organizing the conferences (winter schools earlier) on operating systems' theory since 1975. This topic was quite well bound at that time, but recently the number of theoretic fields in connection with operating systems is permanently increasing and becoming common with another fields of computer science too.

The production of operating systems became a routine activity since mini and micro computers spread extensively. The value, however, of the theoretical results on operating systems is much more, for the construction principles and researches having first appeared in connection with operating systems are now in an extensive use in complex application programs containing a large number of components and having centralized or decentralized controls. Moreover, the efficiency of the operating systems of big computers is also dependent on the behaviour and evaluation of application programs.

In our winter schools we put special emphasis on two theoretical subfields. One is the collection of problems having statistical nature. In this field a large number of our papers dealt with optimal scheduling strategies and storage allocation problems. The second one is the field of logical construction problems. This contained several investigations on process synchronization and communication problems as well as correctness proof methodologies for centralized and distributed systems.

Present volume contains the papers presented at the last winter school held in 1980, which was the closing of our first serie. The next conference will be held in 1982 as a start of a new serie.

TARTALOMJEGYZÉK

OPTIMAL CONTROL OF FINITE SOURCE PRIORITY QUEUES WITH COMPUTER SYSTEM APPLICATIONS

D. ASZTALOS

Computer Centre of National Planning Office, Budapest, Hungary

ABSTRACT

The paper deals with the derivation of optimal control rules for finite source queueing systems with preemptive resume service discipline. The three performance measures considered are: server utilization, mean queue length and throughput.

It is indicated how the results on optimal control can be applied to multiprogrammed computer systems and some numerical examples are given.

OPTIMAL CONTROL OF FINITE SOURCE PRIORITY QUEUES WITH COMPUTER SYSTEM APPLICATIONS

D. ASZTALOS

Computer Centre of National Planning Office, Budapest, Hungary

1. INTRODUCTION.

The closed queueing network models may be successfully applied to the analysis of behaviour of multiprogrammed computer systems consisting of one central processor unit /CPU/ and many peripherals. Each job corresponds to one customer and the CPU and the peripherals are represented by the servers. In large configurations it frequently may be assumed that all jobs of a given mix access separate peripherals. In this case all of the peripherals may be represented by one server with infinite capacity. We use that each closed queueing network consisting of two servers, one of which has infinite capacity, is equivalent to a finite source queueing system. /FSQ/ The server of the equivalent system represents the CPU and the finite source corresponds to the separately used peripherals. In computer system applications the number of customers in the above system is referred as level of multiprogramming. In a given computer system the number

of jobs actually executed in the system is constant for a relatively long time-interval. The definition of the optimal priority allocation for each time-interval, when a preemptive-resume discipline is used, has a great practical importance. The optimality should be defined with respect of a well defined performance measure. The performance measures discussed in this paper are: server utilization, average queue length and throughput.

The paper deals with finite source models with exponential structure. This means that the service times and the residence times at the source are mutually independent, exponentially distributed random variables. Let the number of customers be N and designate them by the integers 1,2,..., N. Let the service time parameter be μ_i the residence time parameter be λ_i for customer i,i=1,2,..,N. We consider only the case when for every i the i-th customer has priority over customers of index higher than i, applying the preemptive resume service discipline. Such systems are referred as exponential finite source priority queueing systems /EFSPQ/.

The FSQs are dealt with in general context in [6] considering mainly the priority disciplines. Tomkó gives in [9] a detailed analysis of the busy period of EFSPQ model. He shows in the case N=2, that the server utilization in steady state case is maximised when $\lambda_1 > \lambda_2$. For the

case N > 2, there are some particular results in [1,2] with respect of optimization of the server utilization. The special conditions of optimality in [1,2] are: $\mu_1 >> \mu_2 >> \ldots >> \mu_N$ and $\lambda_1 >> \lambda_2 >> \ldots >> \lambda_N$. These assumptions are not very realistic in computer systems. Smith proves in [8] the optimality of the $\lambda_1 < \lambda_2 < \ldots < \lambda_N$ control, minimising the mean queue length in steady-state case when $\mu_i = \mu_1$, $i = 2, \ldots, N$.

In this paper the following new results are proved for mean values in steady-state case. The maximal mean value of server utilization in an EFSPQ is achieved when $\lambda_1 > \lambda_2 > \ldots > \lambda_N$. An immediate consequence of this fact is that if $\lambda_i = \lambda_1$, $i = 2, \ldots, N$, then the mean value of server utilization is the same for any priority allocation. But in this case, the throughput is maximal and the mean queue length is minimal if $\mu_1 > \mu_2 > \ldots > \mu_N$. Finally, let assume, that $\mu_i = \mu_1$, $i = 2, \ldots, N$ in EFSPQ. Then the mean value of utilization and throughput are maximal if $\lambda_1 > \lambda_2 > \ldots > \lambda_N$. There are no similar simple rules of optimal control for mean queue length and throughput of EFSPQ when μ_i and λ_i, $i = 1, \ldots, N$ are different.

We note that our results cannot be applied to finite source queueing systems with general service time distributions and nonpreemptive priority discipline unlike to the infinite source models (see e.g. [6]).

2. THE PERFORMANCE MEASURES.

Let us consider an EFSPQ model described in the introduction. We consider three performance measures of an EFSPQ model: mean values of server utilization, queue length and throughput. All these values are examinded in the steady-state of the system.

Let $E\delta^{(j)}$ denote the mean value of the busy period when there are j customers in the system. It follows from the exponential structure that the mean value of the idle period equals to $m_N^{-1} = (\lambda_1+\lambda_2+\ldots+\lambda_N)^{-1}$ when there are N customers in the system. The mean value of server utilization can be calculated as:

$$\rho = \frac{E\delta^{(N)}}{E\delta^{(N)} + m_N^{-1}} \qquad (1)$$

To calculate $E\delta^{(N)}$ we can use the recursive formulas of [9], which are the following:

$$E\delta^{(j)} = \frac{m_{j-1}}{m_j}\left[E\delta^{(j-1)} + \frac{1}{\mu_j}(1-\varphi_{j-1}(\lambda_j))(1+m_{j-1}E\delta^{(j-1)})\right] \qquad (2)$$
$$+ \frac{\lambda_j}{m_j}\cdot\frac{1}{\mu_j}(1+m_{j-1}E\delta^{(j-1)}),$$

$$\varphi_j(s) = \frac{m_{j-1}}{m_j}\left[\varphi_{j-1}(s+\lambda_j) + \frac{\mu_j[\varphi_{j-1}(s)-\varphi_{j-1}(s+\lambda_j)]}{\mu_j+s+m_{j-1}(1-\varphi_{j-1}(s))}\right] \qquad (3)$$
$$+ \frac{\lambda_j}{m_j}\,\frac{\mu_j}{\mu_j+s+m_{j-1}(1-\varphi_{j-1}(s))},$$

where $j = 1, \ldots, N$; and $m_j = \lambda_1 + \ldots + \lambda_j$, and $\varphi_j(s)$ is the Laplace-transform of the busy period variable when there are j customers in the system. These formulas will be used in a slightly different form in the sequel. They are:

$$E\delta^{(j)} = E\delta^{(j-1)} \frac{m_{j-1}}{m_j \mu_j} (\mu_j + m_j - m_{j-1}\varphi_{j-1}(\lambda_j)) + \frac{1}{m_j \mu_j} (m_j - m_{j-1}\varphi_{j-1}(\lambda_j)), \quad (4)$$

$$\varphi_j(s) = \frac{m_{j-1}\left[\mu_j \varphi_{j-1}(s) + (s + m_{j-1})\varphi_{j-1}(s+\lambda_j) - m_{j-1}\varphi_{j-1}(s)\varphi_{j-1}(s+\lambda_j)\right] + \lambda_j \mu_j}{m_j (s + \mu_j + m_{j-1}(1 - \varphi_{j-1}(s))} . \quad (5)$$

One should maximise $E\delta^{(N)}$ to maximise ϱ.

Let Eq be the mean value of the queue length in steady-state, and Eq_j be that part of Eq which corresponds to the customer of priority j. Eq_j is given by the formula (see [6] Ch. IV.) :

$$Eq_j = 1 - \frac{\mu_j}{\lambda_j}(e_{j-1} - e_j), \quad (6)$$

where

$$e_j = (1 + m_j E\delta^{(j)})^{-1} \quad (7)$$

is the steady-state probability of the idle server when there are j customers in the system. Then $Eq = \sum_{j=1}^{j=N} Eq_j$.

Let T be the throughput of the EFSPQ model, which is defined as the mean value of number of customers serviced in unit time in the steady-state case.

Let r_j be the steady-state probability that the j-th customer is found at the source. Obviously

$$r_j = 1 - Eq_j = \frac{\mu_j}{\lambda_j}(e_{j-1} - e_j).$$

r_j can be interpreted as the average time spent by customer j in the source in each time unit. Thus, the throughput of customer j can be calculated as

$$T_j = \lambda_j r_j = \mu_j (e_{j-1} - e_j).$$

Thus, we get

$$T = \sum_{j=1}^{N} T_j = \sum_{j=1}^{N} \mu_j (e_{j-1} - e_j). \qquad (8)$$

3. OPTIMAL CONTROL PROBLEMS

In this section we will prove some results with respect of optimal control of EFSPQ systems.

THEOREM 1. Let us consider an EFSPQ system. $E\delta^{(N)}$ - the mean value of busy period in steady state case - is maximal when $\lambda_1 > \lambda_2 > \ldots > \lambda_N$.

Proof. It is sufficient to show for any j, $1 \leqq j-1$ and $j \leqq N$, that if $\lambda_{j-1} > \lambda_j$, then

$$E\delta^{(j)}(\lambda_1, \lambda_2, \ldots, \lambda_{j-1}, \lambda_j) > E\delta^{(j)}(\lambda_1, \lambda_2, \ldots, \lambda_j, \lambda_{j-1}),$$

if $\lambda_{j-1} < \lambda_j$, then

$$E\delta^{(j)}(\lambda_1, \lambda_2, \ldots, \lambda_{j-1}, \lambda_j) < E\delta^{(j)}(\lambda_1, \lambda_2, \ldots, \lambda_j, \lambda_{j-1})$$

and if $\lambda_{j-1} = \lambda_j$, then

$$E\delta^{(j)}(\lambda_1, \lambda_2, \ldots, \lambda_{j-1}, \lambda_j) = E\delta^{(j)}(\lambda_1, \lambda_2, \ldots, \lambda_j, \lambda_{j-1}).$$

In this proof the priority of customer with parameter λ_i, $i = 1, \ldots, j$ is defined by its position number from left to right in the row rector $(\lambda_{i_1}, \lambda_{i_2}, \ldots, \lambda_{i_j})$.

For the sake of simplicity we use the notations:

$$E\delta_1^{(j)} = E\delta^{(j)}(\lambda_1, \lambda_2, \ldots, \lambda_{j-1}, \lambda_j); \quad E\delta_2^{(j)} = E\delta^{(j)}(\lambda_1, \lambda_2, \ldots, \lambda_j, \lambda_{j-1}).$$

Then, using (4) repeatedly we get:

$$E\delta_1^{(j)} - E\delta_2^{(j)} = \frac{1}{m_j \mu_{j-1} \mu_j} (\alpha\beta - \gamma\varepsilon)(1 + m_{j-2} E\delta^{(j-2)}) \qquad (9)$$

where

$$\begin{aligned} \alpha &= \mu_j + m_j - (m_{j-2} + \lambda_{j-1}) \varphi_{j-1}(\lambda_j), \\ \beta &= \mu_{j-1} + m_{j-2} + \lambda_{j-1} - m_{j-2} \varphi_{j-2}(\lambda_{j-1}), \\ \gamma &= \mu_{j-1} + m_j - (m_{j-2} + \lambda_j) \varphi_{j-1}(\lambda_{j-1}), \\ \varepsilon &= \mu_j + m_{j-2} + \lambda_j - m_{j-2} \varphi_{j-2}(\lambda_j). \end{aligned} \qquad (10)$$

Using (5) we get:

$$\varphi_{j-1}(\lambda_j) = \frac{\mu_{j-1}[\lambda_{j-1} + m_{j-2} \varphi_{j-2}(\lambda_j)] + m_{j-2} \varphi_{j-2}(\lambda_{j-1} + \lambda_j) \cdot D}{(m_{j-2} + \lambda_{j-1})[\mu_{j-1} + \lambda_j + m_{j-2} - m_{j-2} \varphi_{j-2}(\lambda_j)]}, \qquad (11)$$

where $D = m_{j-2} + \lambda_{j-1} - m_{j-2} \varphi_{j-2}(\lambda_{j-1})$,

$$\varphi_{j-1}(\lambda_{j-1}) = \frac{\mu_j[\lambda_j + m_{j-2} \varphi_{j-2}(\lambda_{j-1})] + m_{j-2} \varphi_{j-2}(\lambda_{j-1} + \lambda_j) \cdot C}{(m_{j-2} + \lambda_j)[\mu_j + \lambda_{j-1} + m_{j-2} - m_{j-2} \varphi_{j-2}(\lambda_{j-1})]}, \qquad (12)$$

where $C = m_{j-2} + \lambda_j - m_{j-2} \varphi_{j-2}(\lambda_j)$.

The sign of (9) is defined by the sign of $\alpha\beta - \gamma\varepsilon$.

We show that $\alpha\beta - \gamma\varepsilon$ is a symmetric expression with respect of μ_{j-1} and μ_j, and

$$\alpha\beta - \gamma\varepsilon = \begin{cases} > 0 & \text{if } \lambda_{j-1} > \lambda_j \\ = 0 & \text{if } \lambda_{j-1} = \lambda_j \\ < 0 & \text{if } \lambda_{j-1} < \lambda_j \end{cases}. \qquad (13)$$

This will prove our theorem.

Using (10), (11) and (12) we get:

$$\alpha\beta-\gamma\varepsilon=\left[\mu_j+m_j-(m_{j-2}+\lambda_{j-1})\varphi_{j-1}(\lambda_j)\right](\mu_{j-1}+D)$$
$$-\left[\mu_{j-1}+m_j-(m_{j-2}+\lambda_j)\varphi_{j-1}(\lambda_{j-1})\right](\mu_j+C)$$
$$=\mu_j(D-m_j)+\mu_{j-1}(m_j-C)+m_j(D-C)$$
$$-(m_{j-2}+\lambda_{j-1})\varphi_{j-1}(\lambda_j)(\mu_{j-1}+D)+(m_{j-2}+\lambda_j)\varphi_{j-1}(\lambda_{j-1})(\mu_j+C)$$
$$=\mu_{j-1}y-\mu_j X+m_j(D-C)-\frac{(\mu_j+D)(\mu_{j-1}y+m_{j-2}E\cdot C)}{\mu_{j-1}+C}$$
$$+\frac{(\mu_j+C)(\mu_j X+m_{j-2}E\cdot D)}{\mu_j+D}=\left[(\mu_{j-1}+C)(\mu_j+D)\right]^{-1}*$$
$$*\Big[(\mu_{j-1}y-\mu_j X)(\mu_{j-1}+C)(\mu_j+D)-(\mu_j+D)(\mu_{j-1}+D)(\mu_{j-1}y+m_{j-2}E\cdot C)$$
$$+(\mu_{j-1}+C)(\mu_j+C)(\mu_j X+m_{j-2}E\cdot D)+m_j(D-C)(\mu_{j-1}+C)(\mu_j+D)\Big],$$

where

$$X=m_j-D=\lambda_j+m_{j-2}\varphi_{j-2}(\lambda_{j-1}),$$
$$y=m_j-C=\lambda_{j-1}+m_{j-2}\varphi_{j-2}(\lambda_j),$$
$$E=\varphi_{j-2}(\lambda_{j-1}+\lambda_j).$$

It is easy to check, that $(\mu_{j-1}+C)(\mu_j+D)$ is positive and is the same for both priority orders $(\lambda_1,\lambda_2,\dots,\lambda_{j-1},\lambda_j)$ and $(\lambda_1,\lambda_2,\dots,\lambda_j,\lambda_{j-1})$. Thus, it is sufficient to prove /13/ for $(\alpha\beta-\gamma\varepsilon)^*=(\alpha\beta-\gamma\varepsilon)(\mu_{j-1}+C)(\mu_j+D)$.

Then

$$(\alpha\beta-\gamma\varepsilon)^*=\mu_{j-1}\mu_j(C-D)(X+y-m_j-m_{j-2}E)+\mu_{j-1}D(C-D)(y-m_j)$$
$$+\mu_j C(C-D)(X-m_j)+CD(C-D)(m_{j-2}E-m_j).$$

It is easy to check, that $D(y-m_j)=C(X-m_j)=-CD$.

Thus, with replacements we get:

$$(\alpha\beta-\gamma\varepsilon)^*=(D-C)\Big[\mu_{j-1}\mu_j m_{j-2}(1+\varphi_{j-2}(\lambda_{j-1}+\lambda_j)-\varphi_{j-2}(\lambda_{j-1})-$$
$$-\varphi_{j-2}(\lambda_j))+(\mu_{j-1}+\mu_j)CD+CD(\lambda_{j-1}+\lambda_j+m_{j-2}(1-\varphi_{j-2}(\lambda_{j-1}+\lambda_j)))\Big]. \quad (14)$$

Thus, the $(\alpha\beta-\gamma\varepsilon)^*$ is a symmetric expression with respect of μ_{j-1} and μ_j . It follows from the properties of the Laplace-transform that $1+\varphi_{j-2}(\lambda_{j-1}+\lambda_j)-\varphi_{j-2}(\lambda_{j-1})-\varphi_{j-2}(\lambda_j)>0$, $C>0$, $D>0$ and $0<1-\varphi_{j-2}(\lambda_{j-1}+\lambda_j)<1$.

Furthermore

$$D-C=\lambda_{j-1}-\lambda_j-m_{j-2}(\varphi_{j-2}(\lambda_{j-1})-\varphi_{j-2}(\lambda_j))=\begin{cases}>0 & \lambda_{j-1}>\lambda_j\\ =0 \text{ if } & \lambda_{j-1}=\lambda_j\\ <0 & \lambda_{j-1}<\lambda_j\end{cases}. \quad (15)$$

Thus the sign of $(\alpha\beta-\gamma\varepsilon)^*$ is defined by D-C.

(13) follows from (14) and (15) , and this proves our theorem.

COROLLARY 1. Let $\lambda_1=\lambda_2=\ldots=\lambda_N$. Then $E\delta^{(N)}$ is the same for any priority allocation.

We now consider the EFSPQ systems with homogeneous source, i.e. $\lambda_1=\ldots=\lambda_N=\lambda$.

THEOREM 2. Let us consider an EFSPQ system with homogeneous source. The Eq is minimal and T is maximal when $\mu_1>\mu_2>\ldots>\mu_N$.

Proof. For EFSPQ with homogeneous source $\lambda_i=\lambda$, $i=1,\ldots,N$.

Using (6) and (8) we get for the mean queue length and throughput the following expressions:

$$Eq=\sum_{j=1}^{N}Eq_j=N-\frac{1}{\lambda}\sum_{j=1}^{N}\mu_j(e_{j-1}-e_j)$$

and

$$T=\sum_{j=1}^{N}\mu_j(e_{j-1}-e_j).$$

Thus, $Eq=N-T/\lambda$ and if T is maximised, at the same

time Eq is minimised.

It is sufficient to show that

$$(T_j + T_{j+1})_{\mu_j > \mu_{j+1}} > (T_j + T_{j+1})_{\mu_j < \mu_{j+1}} . \qquad (16)$$

Let us consider the sum

$$T_j + T_{j+1} = \mu_j e_{j-1} - \mu_{j+1} e_{j+1} - (\mu_j - \mu_{j+1}) e_j . \qquad (17)$$

Replacing $E\delta^{(j)}$ in (7) by the r.h.s. of (4) and repeatedly using (7) we get for e_j:

$$e_j = \frac{\mu_j e_{j-1}}{\mu_j + \lambda[j - (j-1)\varphi_{j-1}(\lambda)]} . \qquad (18)$$

Let $C = \lambda[j-(j-1)\varphi_{j-1}(\lambda)]$. It is obvious that $C \geq 0$ for $\lambda \geq 0$. Using (18) the expression (17) may be rewritten as:

$$T_j + T_{j+1} = \mu_j \mu_{j+1} \left(e_{j-1} \frac{\mu_{j+1} + C}{\mu_{j+1}(\mu_j + C)} - e_{j+1} \frac{1}{\mu_j} \right). \qquad (19)$$

It follows from (7) that $e_{j-1} > e_{j+1}$. e_{j-1} does not depend on μ_j, μ_{j+1} and e_{j+1} would not change its value when the customers of priorities j and j+1 are mutually replaced by each other /see COROLLARY 1./. At the same time, if $\mu_j > \mu_{j+1}$, then $(\mu_{j+1}+C)/(\mu_{j+1}(\mu_j+C)) > 1/\mu_j$ and if $\mu_j < \mu_{j+1}$, then $(\mu_{j+1}+C)/(\mu_{j+1}(\mu_j+C)) < 1/\mu_j$. Thus, (16) and the theorem are proved.

We now consider the EFSPQ systems with homogeneous service, i.e. $\mu_1 = \mu_2 = \ldots = \mu_N = \mu$.

THEOREM 3. Let us consider an EFSPQ system with homogeneous service. The $E\delta^{(N)}$ and T are maximal when $\lambda_1 > \lambda_2 > \ldots > \lambda_N$.

Proof. If the service is homogeneous, we get for T:

$$T = \mu \sum_{j=1}^{N} (e_{j-1} - e_j) = \mu(e_0 - e_N) = \mu(1 - e_N). \qquad (20)$$

It follows from THEOREM 1. that $E\delta^{(N)}$ is maximal when $\lambda_1 > \lambda_2 > \ldots > \lambda_N$. It follows from (7) that actually in this case e_N would be minimal, which means that T is maximal when $\lambda_1 > \lambda_2 > \ldots > \lambda_N$. The theorem is proved.

Some earlier results, concerning the homogeneous source case, are contained in [4]. It is proved, that in an exponential, finite homogeneous source queueing system $E\delta^{(N)}$ is the same for a wide class of service disciplines including the processor sharing one, too. Using the results of [5], a closed form solution can be derived for $E\delta^{(N)}$, both in the homogeneous source and the inhomogeneous source systems with processor sharing discipline. The last result is a significant improvment of the algorithm in [9], calculating $E\delta^{(N)}$.

4. APPLICATIONS AND NUMERICAL RESULTS

We have defined some optimal control rules for M/M/1/N /or EFSPQ /queueing systems in the previous section. For the case, when the source is homogeneous, the optimal control rule, minimising Eq and maximising T is the same as for the infinite source M/M/1 preemptive and M/G/1 nonpreemptive systems. This is the $\mu_1 > \mu_2 > \ldots \mu_N$ rule where the index is the priority number.

For the general M/M/1/N system we have an optimal control rule only with respect of the utilization factor ρ, which is the $\lambda_1 > \lambda_2 > \ldots > \lambda_N$ rule.

It would be important for applications to define that set of parameters $S = \{[(\mu_1, \lambda_1), \ldots, (\mu_N, \lambda_N)]\}$ for which the $\mu_1 > \mu_2 > \ldots > \mu_N$ rule is optimal with respect of the throughput T. Numerical examples /see Table 1./, measurements and simulation experiments [7] imply that set S is sufficiently large.

Applying the results of Section 3. to multiprogrammed computer systems, the most important deduction is that under exponentiality assumptions the throughput of the system and the utilization of the CPU cannot be optimised in all cases with the same control rule, in contrast to the widely used heuristic control rule $\mu_1 > \mu_2 > \ldots > \mu_N$, used in practice to improve both performance measures.

In [3] is described how the homogeneous source EFSPQ

system was applied to model a multiprogrammed computer system. However, at that time the $\mu_1 > \mu_2 > \ldots > \mu_N$ control rule was used only heuristically in [3] .

Acknowledgments. The author is greatful to Prof. M. Arató and Dr. J. Tomkó for their valuable suggestions and comments.

Table 1. Numerical examples.

Parameters	Priority allocation	ρ	T	E_q
$\mu_i = 1.0$ $\lambda_1 = 0.2$ $i = 1,2,3$ $\lambda_2 = 0.4$ $\lambda_3 = 0.6$	/1,2,3/	0.690^-	0.690^-	1.076^+
	/3,2,1/	0.720^+	0.720^+	1.217^-
$\mu_1 = 10.0$ $\lambda_i = 0.25$ $\mu_2 = 1.0$ $i = 1,2,3$ $\mu_3 = 0.1$	/1,2,3/	0.810	0.501^+	0.994^+
	/3,2,1/	0.810	0.255^-	1.981^-
$\mu_1 = 1.0$ $\lambda_1 = 0.1$	/2,3,1/	0.856^+	1.182	1.577^-
$\mu_2 = 2.0$ $\lambda_2 = 3.0$	/3,2,1/	0.835	1.936^+	1.375
$\mu_3 = 3.0$ $\lambda_3 = 1.5$	/1,3,2/	0.795	1.810	1.207^+
$\mu_1 = 4.0$ $\lambda_1 = 2.0$	/1,2,3/	0.975	1.803^+	1.408
$\mu_2 = 3.0$ $\lambda_2 = 0.2$	/3,1,2/	0.996^+	0.640	2.746^-
$\mu_3 = 0.5$ $\lambda_3 = 10.0$	/2,1,3/	0.975^-	1.766	1.391^+
$\mu_1 = 2.0$ $\lambda_1 = 0.5$	/1,2,3/	0.506^+	0.924^+	0.708
$\mu_2 = 1.8$ $\lambda_2 = 0.4$	/3,1,2/	0.501	0.908	0.708^+
$\mu_3 = 1.6$ $\lambda_3 = 0.3$	/3,2,1/	0.499^-	0.900^-	0.708
$\mu_1 = 2.0$ $\lambda_1 = 0.5$	/2,3,1/	0.533	1.182	0.757^+
$\mu_2 = 2.2$ $\lambda_2 = 0.4$	/3,1,2/	0.539^+	1.194^+	0.780
$\mu_3 = 2.4$ $\lambda_3 = 0.7$	/3,2,1/	0.537	1.193	0.767
$\mu_1 = 10.0$ $\lambda_1 = 1.0$	/1,2,3/	0.987	1.506^+	1.692^+
$\mu_2 = 1.0$ $\lambda_2 = 1.5$	/2,3,1/	0.987^+	0.681	2.530
$\mu_3 = 0.1$ $\lambda_3 = 1.4$	/3,2,1/	0.987	0.187^-	2.857^-

\+ denotes the optimal value of the measure

\- denotes the worst value of the measure

REFERENCES

[1] Arató M., Diffusion Approximation for Multiprogrammed Computer Systems. Comp.and Math.with Appls.1/3-4/, 314-327 /1975/.

[2] Arató M., Knuth E., Tőke P., On Stochastic Control of Multiprogrammed Computer Based on a Probabilistic Model. IFAC Symp. on Stoch.Control, Budapest, 3o5-311 /1974/.

[3] Asztalos D., A Hybrid Simulation/Analytical Model of a Batch Computer System. Performance of Computer Systems. M.Arató, A.Butrimenko,E.Gelenbe /eds./. North-Holland 149-159 /1979/.

[4] Asztalos D., Finite Source Queueing Systems and Their Applications to Computer Systems /In Hungarian/, Alk.Mat.Lapok /to appear/.

[5] Baskett,F., Chandy, K.M., Muntz, R.R.,and Palacios, G.F., Open, Closed and Mixed Networks of Queues with Different Classes of Customers. JACM 22,2,248-260 /1975/

[6] Jaiswal,N.K., Priority Queues, Academic Press /1968/.

[7] Sherman,S., Baskett, F., Browne, J.C., Trace-Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System. CACM 15,12, 1o63-1o69 /1972/.

[8] Smith, A.J., Multiprocessor Memory Organization and Memory Interference . CACM, 2o,1o, 754-761 /1977/.

[9] Tomkó J., Processor Utilization Study, Comp. and Math. with Appls. 1/3-4/, 337-344 /1975/.

SOME RESULTS ON PETRI NET LANGUAGES

Anna Bagyinszki
L. Eötvös University, Dept. of Coump. Sci.

In this paper we investigate generalized labelled Petri nets introduced in [1] by J.L.Peterson with a slight modification. First we collect the most important definitions.

A Petri net PN is a 5. tuple defined by $PN = (P, T, \Sigma, S, F)$, where P is a finite nonempty set of places, T is a finite nonempty set of transitions, $S \subseteq P$ is the set of start places, $F \subseteq P$ is the set of final places, Σ is a finite set of labels, i.e. an alphabet for labelling the elements of T. Each transition $t_j \in T$ is an ordered triple of the form (σ_j, I_j, O_j) where $\sigma_j \in \Sigma$, I_j and O_j are bags over P (e.g. a place can occur more times in the sequence). I_j and O_j note the input places and output places respectively, for t_j. (A usual representation of a Petri net is a bipartite directed graph. The places and transitions are represented as nodes in the graph, but to distinguish them, place-nodes are represented by circles and transition-nodes by bars. For each occurence of a p_k in the output bag O_j, there is an arc leading from a transition t_j to the place p_k, and for each occurence of a p_k in the input bag I_j, there is an arc leading from the

place p_k to a transition t_j.) The places of a Petri net can be marked, even repeatedly, by putting "tokens", namely, dots in the circles of the graph representation of the net. The so-called execution rule describes the change of the markings of a net under the firing of transitions.

A transition is enabled if all of its input places have (a sufficient number of) tokens in them. A transition fires by removing tokens from all of its input places and placing tokens in all of its output places. The execution of a Petri net begins with one token in a start place. A Petri net may halt whenever it reaches a final state (one token in a final place and zero token elsewhere). Each separate execution of a net defines, or is defined by, the sequence of transitions which are fired during the execution.

We can study the strings obtained from firing sequences by replacing each transition occurence by the corresponding transition occurence by the correspondig transition label. A sequence of symbols which corresponds to an execution starting from a start state and reaching a final state is a computation sequence. The set of the computation sequences of a Petri net is the language of the net, and the class of Petri net languages is denoted by CSS.

Peterson investigated the relationship between CSS and the classes of regular, context-free (CF) and context--sensitive (CS) languages. He also proved the simation of

the class of bounded context-free languages and gave an example for a CF language not being in CSS.

His conclusions are represented in Figure 1.

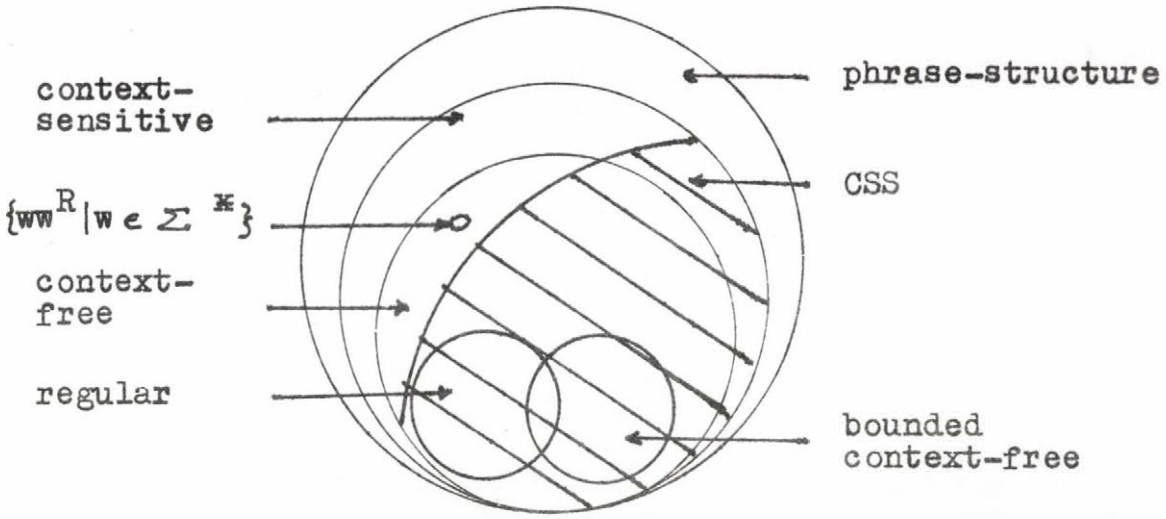


Figure 1.

We extend Peterson's investigations towards linear languages. As it is known, a CF grammar $G = (V_N, V_T, S, P)$ is linear iff each rule is of the form $A \rightarrow PBR$ or $A \rightarrow P$, where $A, B \in V_N$ and $P, R \in V_T^*$. A language L is said to be linear if there exists a linear grammar G such that $L(G) = L$.

The wellknown relationship between linear languages and the classes of regular, CF and metalinear languages can be seen in Figure 2.

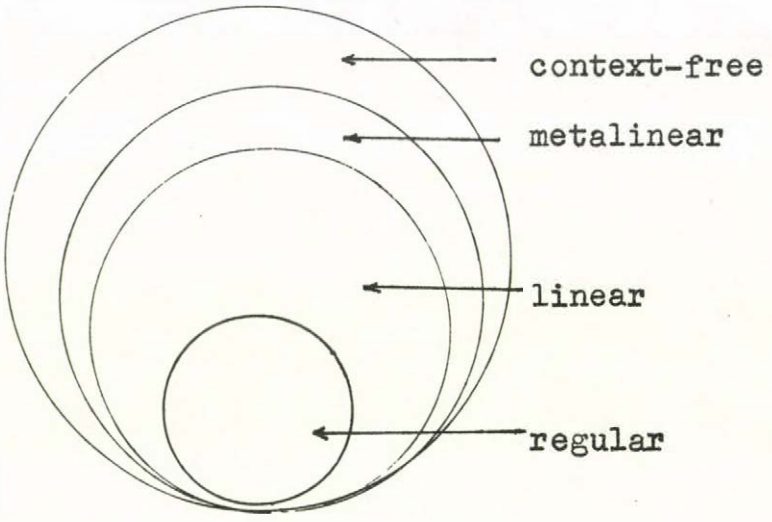


Figure 2.

The question arises: Are all the linear languages in CSS? The answer is no. For example: the language of even-length palindromes $\{ww^R \mid w \in \Sigma^*\}$ in Figure 1. is a linear language but not CSS.

We try to learn a little more about CSS with a method based on grammars.

We need the following definitions:

A linear grammar $G = (V_N, V_T, S, P)$ is said to be type T if the set of productions P does not contain a subset of the form

$$A_1 \longrightarrow P_1 A_2 R_1, \ldots, A_s \longrightarrow P_s A_{s+1} R_s, \ldots, A_r \longrightarrow P_r A_1 R_r,$$
$$A_s \longrightarrow P_s' A_{s+1} R_s' \quad ,$$

where $R_s \neq R_s'$, $A_i \in V_N$, P_i, R_i, P_s', $R_s' \in V_T^*$, $i = 1, \ldots, r$.

A language L is said to be T-type if there exists a type T grammar G such that $L(G) = L$. We will show that the type T languages are in CSS. For this purpose

we want to find an algorithm which construct a Petri net to a type T grammar, which generates the same language.

Theorem: For every language generated by a type T linear grammar we can find a Petri net consisting of $(2n + r)$ places and $(m + r)$ transitions, accepting the language.

Algorithm:

Step 1. Consider $(2n + r)$ places. Label them with symbols $A_1, A_2, \ldots, A_n, Q_1, Q_2, \ldots, Q_n, W_1, W_2, \ldots, W_r$. Let $q = 1$.

Step 2. Take the q-th production rule, it has the form $A_j \longrightarrow P_q A_\ell R_q$ or $A_j \longrightarrow P_q$.

Join an output transition with the label P_q to the place A_j.

Step 3. If there is no nonterminal in the q-th rule, then let Q_j the output place of the transition P_q. Continue at step 6.

Step 4. If the q-th rule has a nonterminal, then let A_ℓ and W_q the two output places of the transition P_q.

Step 5. Join an output transition with the label R_q to W_q. This transition has another input place: Q_ℓ. Its output place is Q_j.

Step 6. Let $q = q + 1$. If $q \leq m$ then continue at step 2. Stop otherwise.

For every rule $A_j \longrightarrow P_q$ the algorithm generates a transition $t_q = (P_q, \{A_j\}, \{Q_j\})$. Firing of transitions

of this type does not change the number of tokens in the net.

For every rule like $A_j \longrightarrow P_q A_\ell R_q$ the algorithm generates a pair of transitions

$t_q = (P_q, \{A_j\}, \{A_\ell, W_q\})$, $t_q' = (R_q, \{W_q, Q_\ell\}, \{Q_j\})$.

After the firing of transitions with label P_q, a new token arises in the net, and after firing of transitions with label R_q, one token vanishes from the net.

W_q is controlling the appearance of the strings P_q and R_q in a word. Their transitions must be fired the same number of times.

Q_i guarantees that the pair of strings appears correctly in the word.

In the net generated by the algorithm the start place is A_1 and the final place is Q_1.

We will show that every word p accepted by this Petri net has a derivation in G. Let us consider the transition sequence $y = t_{i_1} t_{i_2} \ldots t_{i_n}$. The execution corresponding to it starts from the start place A_1 and halts at the final place Q_1. Let $p = U_1 U_2 \ldots U_n$ be the computation sequence assigned to y. After having one token in the place A_1 all the transitions with the only input place A_1 are enabled. The transition t_{i_1} with the label $U_1 = P_{i_1}$ is among them, corresponding to the production rule $A_1 \longrightarrow P_{i_1} A_{\ell_1} R_{i_1}$. After the firing of t_{i_1} one token arises in A_{ℓ_1} and W_{i_1} each, and one token vanishes from A_1.

At the following step those transitions are enabled, which have the only input place A_{ℓ_1}, as transition R_{i_1} with an input place W_{i_1} has another input place Q_{ℓ_1}, which has no token on it. At the k-th step, it is t_{i_k} with the label U_k which is to be fired. We can distinguish three case now:

(a) $U_k = P_{i_k}$ for a rule $A_{j_k} \longrightarrow P_{i_k}$,

(b) $U_k = P_{i_k}$ for a rule $A_{j_k} \longrightarrow P_{i_k} A_{\ell_k} R_{i_k}$,

(c) $U_k = R_{i_k}$ for a rule $A_{j_k} \longrightarrow P_{i_k} A_{\ell_k} R_{i_k}$.

In case (a) one token vanishes from A_{j_k} and one token arises in Q_{j_k}. Because of this the transition $t_{i_{k+1}}$ $(= t'_{i_{k-1}})$ has a token in its second input place too. The input place $W_{i_{k-1}}$ of $t_{i_{k+1}}$ has a token because of the earlier firing of $t_{i_{k-1}}$.

In case (b) one token vanishes from A_{j_k} and one arises in A_{ℓ_k} and W_{i_k}. Now $t_{i_{k+1}}$ with label $U_{k+1} = P_{i_{k+1}}$ is enabled.

In case (c) one token vanishes from W_{i_k} and Q_{ℓ_k}, and one token arises in Q_{j_k}. Now $t_{i_{k+1}}$ with label $U_{k+1} = R_{i_s}$ and with input places W_{i_s} and Q_{j_s} $(s < k-1)$ is enabled. It is clear, that in case (c) we do not use new production rules, we consider only the right sides of earlier ones.

After firing t_{i_n} with label U_n we have only one token

in Q_1 and no one elsewhere. It is necessary that $U_n = R_{i_1}$ for the rule corresponding to the first firing transition t_{i_1}.

It is easy to see that the number of transitions participating in the derivation of word p correspondig to the transition sequence y is n. The same transition can occur several times in the sequence. The algorithm assigns (n+1)/2 production rules to the n transitions.

Thus every transition sequence y has a derivation

$$(A_1 =) A_{j_1} \longrightarrow P_{i_1} A_{\ell_1} R_{i_1},\ (A_{\ell_1} =) A_{j_2} \longrightarrow P_{i_2} A_{\ell_2} R_{i_2},\ \ldots$$

$$\ldots,\ (A_{\ell_{\frac{n-1}{2}}} =) A_{j_{\frac{n+1}{2}}} \longrightarrow P_{i_{\frac{n+1}{2}}}.$$

Conversely, every $p \in L(G)$ has a transition sequence y corresponding to it, which starts at a start state and halts at a final state.

Let $p = P_{i_1} P_{i_2} \ldots P_{i_{n-1}} P_{i_n} R_{i_{n-1}} \ldots R_{i_1}$. The word p is in L(G) if and only if p has a derivation in G. Assume that the derivation has the following form:

$$(A_1 =) A_{i_1} \longrightarrow P_{i_1} A_{\ell_1} R_{i_1},\ (A_{\ell_1} =) A_{j_2} \longrightarrow P_{i_2} A_{\ell_2} R_{i_2},\ \ldots$$

$$\ldots,\ (A_{\ell_{n-2}} =) A_{j_{n-1}} \longrightarrow P_{i_{n-1}} A_{\ell_{n-1}} R_{i_{n-1}},\ (A_{\ell_{n-1}} =) A_{j_n} \longrightarrow P_{i_n}.$$

Using the first production rule we put a token in the start place A_1 of the Petri net. Among the enabled transitions we can find t_{i_1} with label P_{i_1} too. After the firing of t_{i_1}, only A_1 and W_{i_1} will have token (s).

Those transitions are enabled which have the only input place A_{ℓ_1}. The transition t_{i_2} with label P_{i_2} is among them. Using the (n-1) -th rule we have a token in the place $A_{j_{n-1}}$ and so the transition $t_{i_{n-1}}$ with label $P_{i_{n-1}}$ is enabled. After the firing of $t_{i_{n-1}}$ we have tokens in the places $A_{\ell_{n-1}} = A_{j_n}$ and $W_{i_{n-1}}$. Then the transition t_{i_n} with label P_{i_n} is enabled. Its only output place is Q_{j_n}. Having a token in Q_{j_n}, the transition $t'_{i_{n-1}}$ with label $R_{i_{n-1}}$ is enabled. Its input place $W_{i_{n-1}}$ hase got a token at the (n-1) -th step. The firing of $t'_{i_{n-1}}$ puts a token in the place $Q_{j_{n-1}}$ and so the transition with label $R_{i_{n-2}}$ is enabled. Lastly the firing of t'_{i_1} puts the last token in the final place Q_1.

Thus we can assign the transition sequence

$y = t_{i_1} t_{i_2} \ldots t_{i_{n-1}} t_{i_n} t'_{i_{n-1}} \ldots t'_{i_1}$ of the Petri net to the word

$p = P_{i_1} P_{i_2} \ldots P_{i_{n-1}} P_{i_n} R_{i_{n-1}} \ldots R_{i_1}$ in $L(G)$,

so the theorem is we proved. In this form the algorithm is not applicable for metalinear grammars because of the existence of more than one nonterminal symbols in the right sides of the rules. But it is easy to see that the class of type T metalinear languages is also contained in CSS, as it is proved by Peterson, that CSS is closed under union and

concatenation and it is known, that the metalinear languages can be constructed from linear languages by these two operations.

Figure 3. illustrates the situation of type T languages.

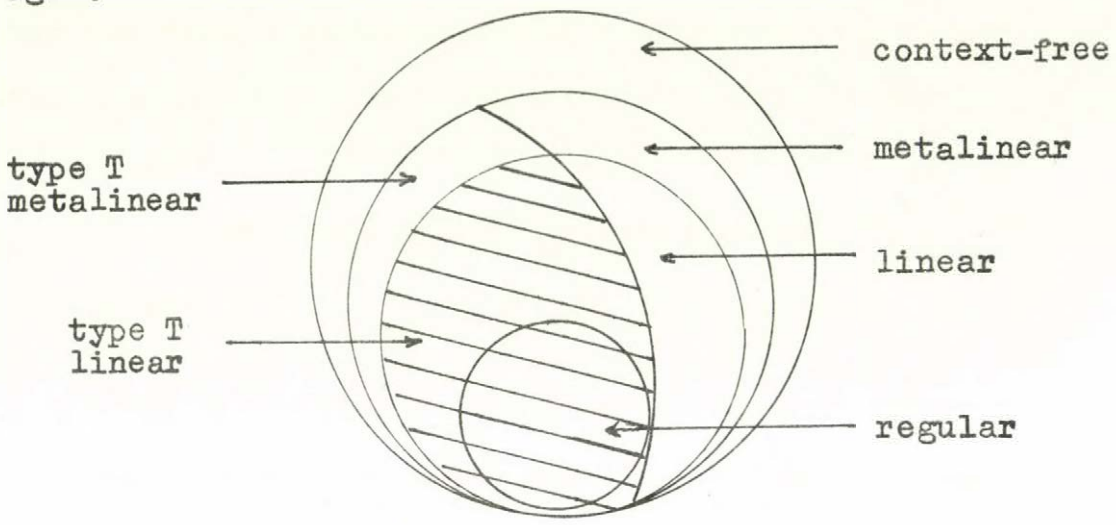


Figure 3.

References

[1] Peterson, J.L.
Computation Sequence Sets
Journal of Computer and System Sciences 13, 1-24, 1976.

QUALITY OF SERVICE MONITORING OF THE OPEN SYSTEMS ARCHITECTURE TRANSPORT LAYER

Jerzy A. Barchański
Tomasz Muehleisen

Abstract: The subject of the paper are problems related to quality of service monitoring of the transport layer within the frame of the Open Systems Architecture. At the beginning, general aspects of performance determination in layered network architectures are presented. It is followed by introduction of a quality of service notion and by definitions of proposed quality of service criteria for the transport layer. As a saquel, measurements tools applicable for the criteria value determination are presented. It is shown as well how to evaluate the criteria. Finally, possible extentions to the measurement program are suggested.

I. INTRODUCTION

Computer networks have become at present distributed data processing systems of considerable complexity and cost and for this reason alone, they require a profound insight into their functional characteristics. Measurement is one of the way commonly used for this purpose and it is especially suitable for performance evaluation of the operational networks. Quanitative data thus obtained allow one to determine if a network behaves according to its design specifications and it is frequently the only way in which design flaws, impossible to be traced at earlier stages of the network development, can be found out and corrected. Other information of utmost importance achievable by measurements is the quality of services provided by the network to its end-users. Those end-users - in layered models of the network architectures - are provided with end--services on top of a chierarchy of partial services provided by consecutive layers to their "up-stairs neighbours". Due to this the general problem of quality of services monitoring can be decomposed into a number of partial problems related to services provided by each individual layer of the architecture.

The problems presented in the paper concern the quality of service monitoring of the transport layer for our inter-university computer network under design [2]. As no specific transport protocol has been designed yet and the network designers have adopted the ISO 7-layered view of the architecture, we have chosen the transport layer within the frame of the Open Systems Architecture [1] as an

object of measurement experiments. Departing from some general considerations, we have shown how for the specified object to quantify its quality of services, what measurement tools to apply and how to evaluate this quality of services provided to the session layer. Furthermore, in compliance with the Open Systems Architecture's philosophy, we have suggested that the quality of services of an individual transport connection can be maintained within pre-determined limits by means of some control mechanisms.

It is our hope that future practical solutions based upon our general-to-a-degree considerations will give a local /in the meaning of architecture, not topology!/ insight into the network operational behavior, possibly resulting in an improvement of the network performance.

II. PARAMETERS OF AN OPEN SYSTEMS ARCHITECTURE LAYER

As we have mentioned the Open Systems Architecture recently introduced by the ISO can be viewed as one to the approaches to problems of design and analysis in computer networks. The architecture itself has been amply documented [1] and needs little introduction. For the needs of the paper we shall present selected parts of it only, concerning problems of performance within the multi-layered architecture. Throughout the presentation we shall use the original ISO terminology.

A basic principle underlying the concept of the Open Systems Architecture is structuring of the network of co-operating systems by its logical division into a /7-element/ set of layers, so that each individual layer can be viewed as a distributed sub-system. Except for the highest layer, each layer /N/ provides the next higher layer /N+1/ with a set of services by performing all its functions and utilizing the most appropriate services available from the next lower layer /N-1/. Within each system the actual provision of services is assured by entities, which for each individual layer co-operate between each other by means of peer-to-peer protocols. The provision of services to entities in the /N+1/th layer takes place upon a request made by the entities of the same layer to entities of the Nth layer, and the request for services is accompanied by a re-

quest to maintain a specified quality of those services. /The latter request is implied, rather than physically present./

The notion of quality of services can be interpreted as a quantitively expressed value of the services, represented either as a value set of some quality criteria, or as a single value of a global quality criterion. Within each of the entities resides a monitoring function and its task is to measure and evaluate those quality of service values and to send their current estimation to the entities in the next higher layer. This process proceeds in a "cascade-like" fashion. /see Fig.1./

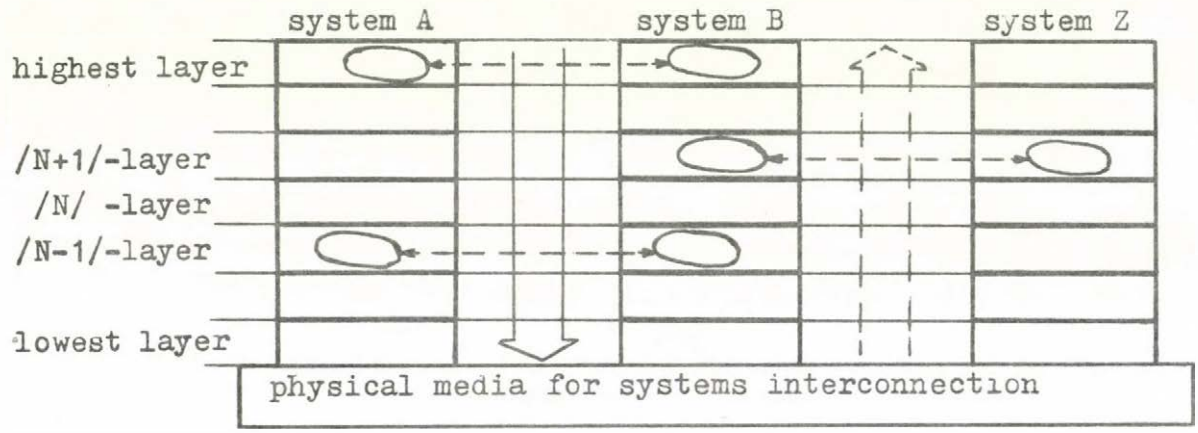


Fig. 1. Requests for services and their fulfilment within the model of the Open Systems Architecture.

It should be noted that at this stage the actual location of the monitoring function is immaterial and will be defined later.

Although we have spoken of the quality of service parameters as of an indication of a layer operation, those parameters are not necesserily the only ones to be cognized. We shall devote a few lines to this more general problem now.

Each layer of the Open Systems Architecture can be characterized by some parameters, the values of which should be tuned - and, perhaps dynamically updated - taking into account the avail-

able information on what is outside the border of the layer. To do this, it is necessary to establish the relations between parameters in the same layer and parameters in the adjacent layers. In establishing those relations, one must distinguish the design phase from the operational phase.

In the design phase:

a/ some parameter values can be assumed by means of a forecast /prediction/,

b/ some others can be assumed as objectives /for example, some parameters that express the level of service provided to the users of a layer/,

c/ some others have to be referred to as constraint values,

d/ other parameter values can be estimated by making use of theory and experience - in order to achieve those objectives /b/ within those constraints /c/ and under those hypotheses /a/.

In the operation phase, measurements can reveal that hypotheses /a/ need to be modified, and the objectives /b/ can be improved or must be reduced, within the constraints /c/ by a different setting of /d/ parameters.

In short:

in the design phase: $d = d(a,b,c)$

in the operational phase: $c = c(a,b,d)$

where both a and c parameters can be measured in the operational phase, whilst they are assumed to be "a priori" in the design phase.

From another viewpoint the characteristic parameters of a layer can be divided into the following four sets:

a/ parameters, defined to protect the transient states of the layer /e.g. time-outs, number of retries of the same action/,

b/ parameters, defined to provide the higher layer with a specified quality of service /e.g. throughput, response time, delay introduced, level of reliability, etc./,

c/ parameters, defined to evaluate the cost of implementing those functions of the layer, which are necessary to provide the layer's services /they may be expressed e.g. in terms of overheads for addresses and control information, either in the header portion of the message-carrying data, or in special messages carrying no data, and invisible to the next higher

layer/,

d/ parameters, defined to make the best use of the network resources in the operation of the functions considered above.

A user of the layer services is interested mainly in the /b/ and /c/ parameters, so we shall deal in the following with these groups only. Nevertheless, it is useful to measure the parameters of the /a/ and /d/ group as well, due to inter-relations which exist between all of them.

III. GENERAL CONCEPTS OF QUALITY OF SERVICE MONITORING

In the previous chapter we have made a remark that the actual location of the monitoring function was at that stage immaterial. The problem of the function location within the multi-layered architecture is one of the open questions for the network designers and should be resolved while the network is still in design stages.

At least two solutions are possible:

/i/ the operation of each entity of a layer is monitored by a local monitoring module implemented as a special procedure in this entity /see Fig. 2a/,

/ii/ the operation of each entity of a layer is monitored by a seperate monitoring module implemented as a special procedure in the entity of the next higher layer /see fig. 2b/.

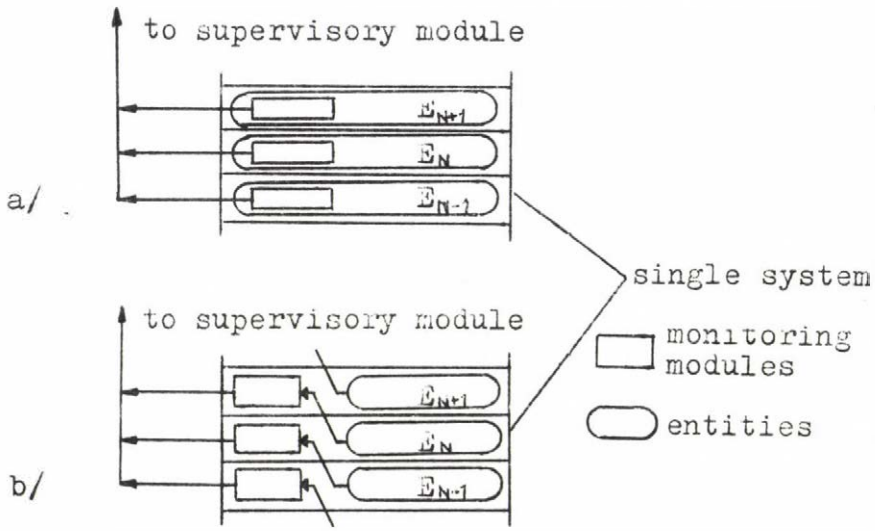


Fig. 2. Alternatives for location of monitoring modules.

For each of the above alternatives again two possibilities exist for the location of a module supervising the operation of all the monitoring modules:

/i/ each layer in the network as a whole can be equipped with the supervisory monitoring module,

/ii/ each system of the network can be equipped with the supervisory monitoring module

As the rest of the paper is devoted to monitoring of a single layer only, the actual locations and connections among all of the monitoring modules are not important for our purposes. Nevertheless, any global solution to problems of monitoring of the network operation should be based upon an appropriate of the monitoring system structure.

IV. TRANSPORT LAYER SERVICES

In this chapter we shall focus our attention on the transport layer, for which the monitoring module should be designed as a part of the monitoring system for the inter-university network [2,3].

The transport layer exists to provide the transport services in association with the underlying services provided by supporting layers. The transport service provides transparent transfer of data between session entities and the transport layer relieves the transport users /resident in the session entities/ from any concern with the detailed way in which reliable and cost effective transfer of data is achieved. For this purpose the transport layer provides a set of services to the session layer and the services themselves can be groupped into the three categories:

/i/ connection-oriented services,

/ii/ transaction-oriented services;

/iii/ broadcast-oriented services.

The transport service is provided by the transport layer performing all necessary functions in conjunction with the utilization of the most appropriate underlying facilities and quality of service available from the supporting layer. The transport layer is required to optimize the use of the available communications resources to provide the performance required by each communicating

transport user at minimum cost.

The transport layer will perform all those functions that are necessary to bridge the gap between the services provided by the network layer and the services needed by the session layer. For this reason the functions performed are dependent on two criteria: the services provided by the underlying network layer and the services required by the session layer. The services provided by the transport layer to the session layer can be segmented into three groups, each associated with the three phases of a transport connection: the establishment phase, the data transfer phase and the termination phase.

The establishment services group consists of the following services:

a/ transport connection establishment,
b/ class of service negotiation,
c/ protocol negotiation.

The data transfer services group consists of:

a/ transport-service-data-unit transfer,
b/ expedited-transport-service-data-unit transfer,
c/ sequencing,
d/ error notification,
e/ purge,
f/ security/encryption,
g/ flow control,
h/ priority.

The transport layer provides at last termination services. The service does not guarantee the delivery of data that proceeds the transport clear to the corresponding user of the transport layer. The transport connection will be terminated regardless of the action teken by the correspondent.

Having described the transport layer and its services as an object of our quality of service monitoring program, we proceed to present a way in which the monitoring will take place.

V. PRINCIPLES OF TRANSPORT LAYER QUALITY OF SERVICE MONITORING

Our considerations concern the monitoring of the transport layer now. The necessity to monitor the layer quality of services is expressed openly in the outline of the Open Systems Architecture's philosophy [1,p.65] in contrast to the lower layers, for which it is only implied.

As said before the transport users are capable of establishing, maintaining and terminating the transport connections by means of transport protocols which support a simultaneous operation of many transport connections. For this reason we project the notion of quality of services from the layer as a whole to the individual transport connections between two entities. For each connection, we shall identify its current quality of service as a set of values of some quality criteria evaluated by the monitoring function. Additionally, we shall identify the requested quality of service as a set of boundary values of the criteria as chosen by the user /i.e. the session layer/ at the transport connection establishment time. This quality of service will be currently monitored and should be maintained throughout the life-time of the transport connection. In other words, any failure to maintain the agreed quality of service on any given connection will necessarily terminate the existance of that connection.

From the above it results that in order to maintain the requested quality of service a control function should exist, capable of comparing the current /i.e. monitored/ and the requested /i.e. negotiated/ qualities of service on each transport connection. If for a connection the former falls below the latter, the control function requests the transport layer to terminate that particular connection. As transport connections are established upon network connections, the terminated connection can be re-established by choosing such /and such a number of/ network connections which give more chances of maintaining the requested quality of service. The Figure 3 attempts to illustrate the above considerations. It should be noted that the actual location of the control and the monitoring function can be different from those depicted without any impact on the presented

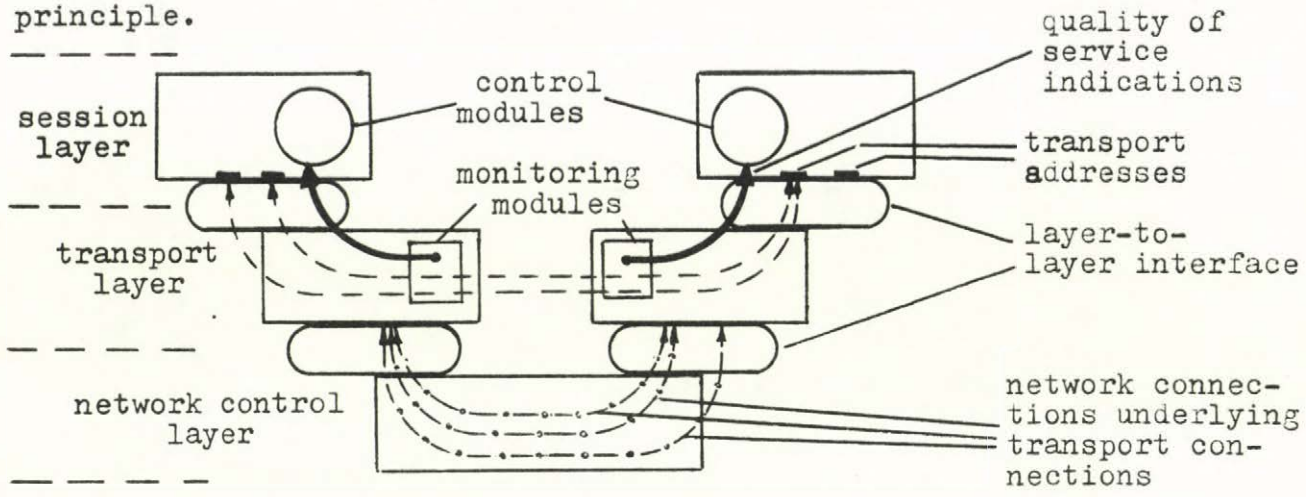


Fig. 3. The monitoring function and the control function for the transport layer.

The presented approach to the issue of the quality of service monitoring within the transport layer can be used in design of any layered realization of a network. Since for the inter-university network higher-layer protocols are in early development stages, the appropriate software can easily be designed to acomodate the monitoring and the control function.

The practical utilization of the above considerations is additionally justified by the fact that in the planned network the network control layer will provide the X.25 service making use of virtual circuits and permanent virtual circuits 4 . Establishment of transport connections upon network connections by performing N:1, 1:N or just 1:1 mapping /the first two referred to as "upward" and "downward" multiplexing, respectively/ can result in a variety of quality of service and cost ranges provided by the individual transport connections. The indication of a current value of the quality of service and the previously registered quality indications /averaged over a period of time/ enable the layer to choose the most cost-effective connections. Then, by means of the feedback loop in the multi-layered network architecture, a local /i.e.one-level/ improvement of performance can practically be achieved.

VI. TRANSPORT LAYER QUALITY OF SERVICE CRITERIA

A proper choice of the quality of service criteria is a fundamental problem both for networks in early design stages and for those already operational. The question what aspects of either predicted or real behavior will be cognizable depends heavily on the chosen set of criteria.

When determining the quality of service criteria for the transport layer, we have departed from a number of asumptions. The basic asumption was that the criteria to be proposed should meet the general requirements of a performance metric, namely they should [5]:

/i/ have a physical meaning or interpretation,
/ii/ be quantitative,
/iii/ be an indicator of the system performance.

Another asumption was that, in compliance with the Open Systems Architecture's philosophy, the criteria should be applicable to all the consecutive phases of the transport connection operation, i.e. to:

/i/ the connection establishment phase,
/ii/ the data transfer phase,
/iii/ the connection termination phase.

Implementation requirements place yet another constraint on the choice of the criteria allowing for the ones observable by software monitoring techniques only.

For the asumptions as above we have proposed the following set of the criteria applicable for the three phases of the transport layer operation.

/i/ for the connection establishment phase:
- connection set-up delay /T_{SET}/,
- connectability /α/.

/ii/ for the data transfer phase:
- mean /maximum/ throughput /β/,
- transit delay /δ/,
- reliability /ϱ/,
- efficiency /γ/,
- availability /ω/.

/iii/ for the connection termination phase:
- connection termination delay.

The above criteria meet all the requirements set forth before in a straightforward fashion.

In compliance with the preliminary project of the inter-university network, its transport layer will support the connection-oriented services /based upon the X.25 services provided by the network control layer/ and the set of the criteria has been chosen accordingly. Should a need arise in the network in the future to accomodate the transaction-oriented services or the broadcast services, another sets of criteria will have to be proposed.

In the following we shall present definitions and physical interpretation for the quality of service criteria.

Connection set-up delay is an indication of how fast the transport connection can be established and it allows the session layer to differentiate the connections according to their /mean/ establishment delay and to choose those for which the delay is minimum. This is especially useful in the conversational mode of the network utilization, when the network users /and consequently transport users/ change frequently their correspondents.

$$T_{set_i} = \frac{1}{N} \sum_{i=1}^{N} (T_{a_i} - T_{b_i})$$

where:

T_{a_i} - time when the session layer issues a request to establish i-th transport connection,

T_{b_i} - time when i-th connection is established,

N - number of espablished connections.

Connectability is an indication of possibility to establish the transport connection, or in other words, it is a probability of a succesful connection establishment.

$$\alpha_i = \frac{N_a - N_b}{N_a}$$

where:

N_a - number of all attempts to establish the i-th transport connection,

N_b - number of failures to establish the i-th transport connection.

Throughput is an indication of the connection's capability to transfer message-carrying data between the transport addresses. The mean value of throughput enables one to determine the transport connection "traffic", whilst the maximum value is an indication of the connection performance limits. Throughput is especially useful for the networks which operate in the mass-data-transfer mode.

$$\beta = \frac{B}{t}$$

where:

B - maximum or averaged over a period of time number of information bits for maximum and mean throughput, respectively,

t - data transfer time.

Transit delay is an indication of how fast transport-data units can travel across the transport connection. It is often more convenient to use so called round-trip delay, i.e. the time elapsed from the moment the transport-data unit was dispatched by the sender, to the moment an acknowledgement of that unit was recieved at the sender's side of the connection. This criterion is especially useful for the networks which operate in the conversational mode.

$$\delta = \frac{1}{N}\sum_{i=1}^{N}(T_a - T_b)$$

where:

T_a - time when the transport-data unit is dispatched,

T_b - time when an acknowlegement of that unit is received at the sender'side,

N - number of the transport-data units sent.

Reliability is an indication of the connection's degree of protection against errors, both recoverable and unrecoverable. As the error recovery within the layer is usually carried out by means of the retransmission mechanism, retransmission rate can be chosen as a reliability criterion for recoverable errors. Hence, asuming the correct operation of the retransmission mechanism:

$$\varrho_1 = \frac{N_a - N_b}{N_a}$$

where:

N_a - number of dispatched transport-data units,

N_b - number of retransmitted transport-data units.

The asumption that the error control mechanism works correctly need not be realistic, and for this reason we have chosen error rate as a complementary reliability criterion. This parameter is useful to characterize situations when an error detected by the error control mechanism of the transport layer can not be recovered /i.e. by retransmission/ by the layer itself and an appropriate action has to be taken at the session layer.

$$\varrho_2 = \frac{N_a}{N_b}$$

where:

N_a - number of error notifications issued by the transport layer,

N_b - total number of transport-data units sent.

Efficiency is an indication of the connection's utilization for transmitting message-carrying data and it is especially useful in reflecting the share of the control data overhead in the whole of the data.

$$\gamma = \frac{D}{N_1(D+H) + N_2 A}$$

where:

D - number of information bits in the transport-data unit,

H - number of control bits in the transport-data unit,

A - number of control bits in the acknowledgement,

N_1 - number of transmissions of the transport-data unit,

N_2 - number of transmissions of the acknowledgement.

Note: the above formula does not hold true for "piggy-backed" acknowledgements.

Availability is an indication of probability that for the successfully established connection, the data transfer phase can proceed until all the data have been transmitted and it is only then, that the connection can be terminated. In other words, this is a probability that the connection will not be terminated /or generally speaking unavailable/ while it is needed for data transfer. Hence,

a precise definition of availability of the connection should be based upon time parameters.

$$\omega = \frac{T_a - T_b}{T_a}$$

where:

T_a - time of availability of the connection,
T_b - time of unavailability of the connection.

Those time parameters however, may be difficult to measure and we have proposed another equivalent definition based upon event parameters.

$$\omega = \frac{N_a - N_b}{N_a}$$

where:

N_a - number of succesfully transmitted transport-data units,
N_b - number of untransmitted transport-data units due to unavailability of the connection

Connection termination delay is an indication of how fast the connection can be cleared and it allows the session layer to differentiate the connections according to their /mean/ termination delays. This may be useful in the conversational mode of the network utilization, when the network users /and consequently transport users/ change frequently their correspondents.

$$T_{term} = \frac{1}{N}\sum_{i=1}^{N}(T_{a_i} - T_{b_i})$$

where:

T_a - time when the session layer issues request to terminate the i-th connection,
T_b - time when the i-th connection is terminated,
N - number of terminated connections.

The above presented set of the quality of service criteria can be further extended to accomodate criteria characterizing some other aspects of the transport layer operation. This basic set is however, sufficient to give a many-sided insight into the operation of the transport connections. Out of this criteria base, some quality of service parameters can be selected for the on-line monitoring /i.e. throughput, error rate/.

The insight into the quality of service of the transport connections would not be complete without the cost factor. The knowledge of costs of the transport connections enables the session layer to choose the connections for which a given level of the quality of services can be achieved at minimum cost, or in short, for which the services are most cost-effective.

Since in the inter-university network, the packet-switched network will provide the X.25 services, the cost of the transport connections is dependant on the costs of the network connections underlying those transport connections. Thus:

$$C_{trans_i} = \begin{cases} \sum_{j=1}^{N} C_{net_j} + A & - \text{for the "downward" multiplexing} \\ \frac{C}{N} net_j + A & - \text{for the "upward" multiplexing} \\ C_{net_j} + A & - \text{for the 1:1 mapping} \end{cases}$$

where:

C_{net_j} - cost of the j-th network connection,
C_{trans_i} - cost of the i-th transport connection,
N - number of the X.25-type connections,
A - constant factor.

The constant factor A results from costs of the operation of the appropriate transport stations supporting the transport connection. Its value can be determined by dividing an arbitrary cost of all the software and hardware resources of the transport layer through by the current number of transport connections. /How to extract those resources from the rest of the network is a challenging problem for the network analysts./

As to the cost of the X.25-type network connections, it will be established by the owner of the packet-switched network, i.e. the Post Office Authority. The actual cost of the connections' utilization is dependent on such factors as: connections' throughput, reliability, their lengths /geographic and topological/ and many others. Sometimes however, the charges do not take into account those factors at all, reflecting only a network policy of the authority. In any case, those costs will be known once the packet-switched network is operational.

VI. MEASUREMENT TOOLS

As we have mentioned earlier, measurements in the inter-university network will be carried out exclusively by software methods, which considerably narrows the whole range of the applicable measurement tools [5]. Despite the artifact they introduce, the software methods are commonly used for this purpose, mainly due to their versatility and straightforward implementation. The tools presented in the references [6,8,9] have been applied to the packet-switched communications networks /i.e. the network layer/ only, and we shall here-in review those tools with particular regard to their applicability for measurements of the transport layer.

In a most general sense, the measurement tools can be segmented into three groups:

/i/ observation tools,
/ii/ analysis tools,
/iii/ special-purpose tools.

The observation tools are used for direct data measurements and can be further classified into:

/i/ snap-shot observation,
/ii/ trace observation,
/iii/ accumulated observation,
/iv/ status reports.

Snap-shot observation consists in the data gathering in the network at some pre-determined instants of time. Those time instants should be chosen carefully, so that the measurement data could reflect time changes of the network operation on one hand, and the amount of data should not be excessive placing unnecessary burden on the network, on the other hand. In the network node the subject of the snap-shot observation are: the lengths of the input and output queues, the current state of the routing table, the length of the free storage list and others. This tool can not be applied directly to the monitoring of the quality of service of the transport layer. It can however, be used for determining

an instanteous state of the transport layer resources. In particular, it can be used for measurement of the current number of the transport connections and the number of free buffers.

Trace observation is used for determining the packet route as it crosses the communications network. This is achieved by so-called time-stamps annotated during the time when the packet passes through a sequence of the network nodes. Those time-stamps can reflect, for example times when: /1/ the last bit of the packet arrives, /2/ the packet is put on a queue, /3/ the packet starts transmission and /4/ the acknowledgement is received /for store--and-forward packets sent to a neighbouring node/. As the transport layer protocol supports the end-to-end connections, the trace observation can be utilized by extending its scope to the host computers and/or to their front-ends. With this approach, the time-stamps can be applied for determining all the component values of the transport connection delay by time-stamping a measurement packet as it travels across the transport connection. It should be noted, that the component delays can be calculated from the differences between the time-stamps in the same packet, while throughput from the differences between corresponding time--stamps in succesive packets./ see 3 and / /.

Accumulated observation is a measurement tool for gathering data characterizing activity of the communications network in an interval of time. This activity is summerized in a set of activity tables including such sample data as: the packet sizes, the number of packets, the number of the acknowledgements, the time of sending packets and acknowledgements and many others. This tool can be applied for the transport layer monitoring and the following data should be gathered:

- number of sent and received transport-data units,
- number of sent and received acknowledgements,
- times of sending transport-data units,

times of receiving acknowledgements,
number of retransmissions,
number of error notifications and others.

The above parameters will be used for evaluating the proposed quality of service criteria for the transport connections. The accumulated observation can also result in some histograms characterizing the transport layer operation.

The status reports allow in the communications network to monitor the operational state of its nodes and their line adaptors. The measurement data is transmitted to the sub-network operator, enabling him to undertake appropriate switching activities. This method can easily be transferred to the outside of the communications sub-network, where it can be applied for determining the operational state of the processors /hosts and their front-ends, if any/, line adaptors and terminals. The application of the status reports is not directly connected with the quality of service evaluation. It can however, reveal the real causes of the quality of service degradations in the transport layer.

The above presented methods are applicable for gathering of the measurement data only. As a general rule, those data will have to be further processed and to serve this purpose, the analysis tools are applied. Two methods can be distinguished:

/i/ on-line analysis,
/ii/ off-line analysis.

The usefulness of one or the another of the above methods depends, generally speaking, on the degree of impact the analysis results have on an improvement of the current network behavior.

On-line analysis is useful when a short interval of time is required between the monitoring activities and subsequent changes in the current network characteristics caused by its operators /human or programmable/. When this requirement is met, the network monitoring helps control directly the quality of services provided by the network. We can say that, by analogy to the experiment

control rules, the on-line analysis tool helps control the network operation by means fo a "short" feedback loop.

Off-line analysis is useful for determining the network behavior in a considerably longer period of time and for this purpose, it utilizes mathematical statistics methods. This analysis can however, be applied for controlling the quality of services in the network, but the interval of time between the measurements and subsequent changes in the network functional characteristics is uncomparably longer /here design changes are sometimes necessary rather than re-establishments of the transport connections/. Using the earlier analogy, the off-line analysis helps control the network operation by means of a "long" feedback loop.

The special-purpose tools are applicable for experimental measurements in the network and we shall present briefly here-in the most commonly used tool.

Artificial traffic generators are used for generating a fictitious traffic in the network, totally different from the user-data traffic. The generators are usually applied for experimentally determining the network performance limits, i.e. maximum throughput and for this purpose they create and send as heavy a traffic as the communications sub-network can acomodate.

In conclusion, for the current evaluation of majority of the quality of service criteria the application of the accumulated observation and on-line analysis is sufficient. However, the other tools can be applied in order to gain a more complete insight into the operational state of the transport layer resources.

VII. QUALITY OF SERVICE CRITERIA EVALUATION

In this chapter we shall speak about the ways in which the practical monitoring of the quality of service will take place. In order to show the locations of measurement points, we shall

present below the structure of a single transport entity, commonly referred to as transport station. Bearing in mind that the communications sub-network will provide the X.25 services, the minimum set of modules performing the transport functions is the one illustrated in the Fig. 4.

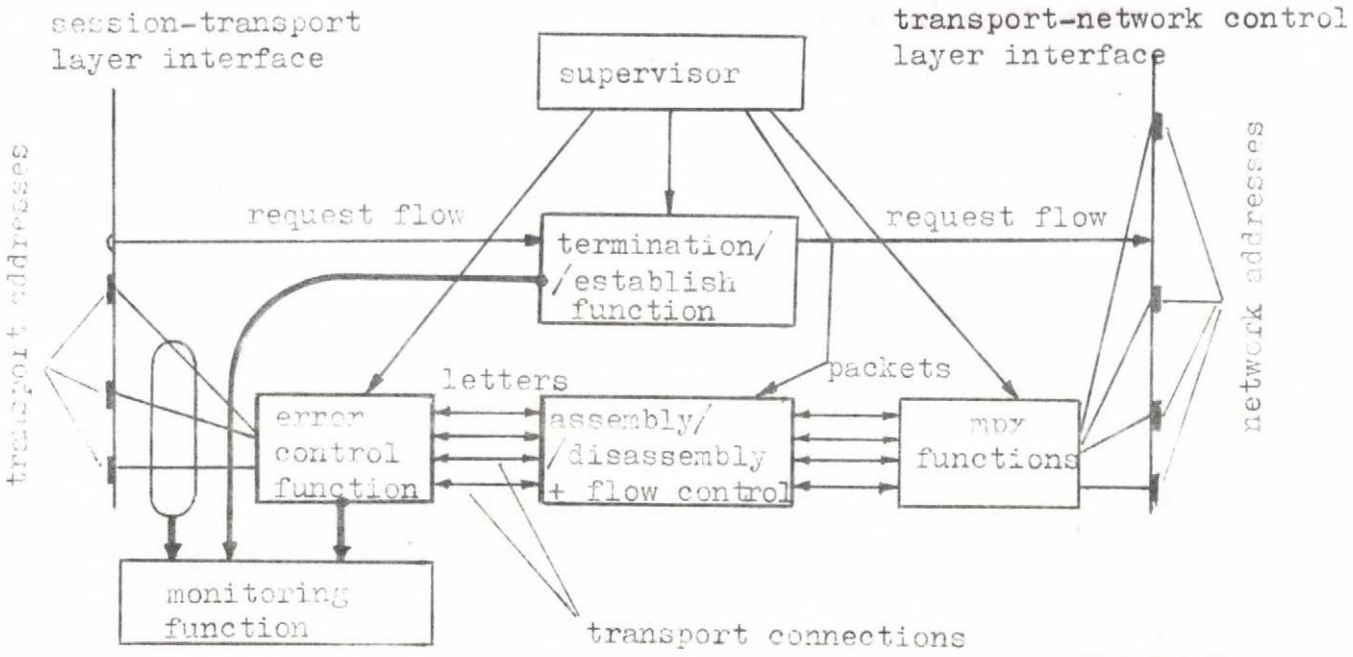


Fig. 4. The transport entity structure and locations of measurement points.

When analysing the quality of service criteria, it can be seen that their component parameters are related to the establishment/ /termination function, the error control function and to the flow of the transport-data units across the transport entity. It should be noted that the location of the monitoring function within the entity as suggested by Fig. 4 is just one of the alternatives for this location /see Chapter III/.

In the following we shall present some hints for practical evaluation of the quality of service criteria.

Connection set-up delay can be evaluated from the differences between the time when the session layer issues SET CONNECTION /or equivalent/

commandand the time when the information is received from the network control layer that all the connections underlying this transport connection have been established. The appropriate time instants are easily observable and if stored over a period of time. Othe average set-up delay of this particular connection or of all the connections /"double" average/ can be calculated.

Connectability can be evaluated by monitoring succeses and failures to establish the transport connection. Hence, the component parameters can be gathered from the establishment/termination function equipped with a timer. Its task is to control the connection establishment time and when this time is excessive /i.e. greater than a specified time-out/, a failure to establish the transport connection can be recorded.

Connection termination delay can be evaluated form the differences between the time when the session layer issues TERMINATE /or equivalent/ command and the time when the information is recieved from the network control layer that it has terminated all the network connections underlying this transport connection. Similiarly, in order to evaluate average delays the time parameters have to be stored over a period of time.

Throughput can be evaluated by monitoring the number of the acknowledged transport-data units over a period of time. Additionally, for each data unit its size should be memorized in order to express throughput in bits/sec. By extending considerably this time period the mean throughput can be measured. In order to determine the maximum throughput, the artificial traffic generators should be used for generating as many transport-data units as the communications network can accept.

It should be noted, that the above considerations are related to throughput evaluation in the networks in which each system is equipped with the measurement module. If only one measurement module is present the throughput can by evaluated by the utilization of special measurements packets or normal data packets with especially interpretable text.

When using the data packets, the times of leaving the transport station should be stamped in the data field and the actual data can follow them on the obvious condition that the data beginning is marked somehow. The throughput can be evaluated from the time differences of consecutive packets.

Transit delay can be evaluated from the difference between the time when the packet is received at its destination and the time when it is sent from the source. It should be noted however, that for practical measurements both the destination and the source transport station clocks would have to be synchronized which is, by no means an easy problem. Due to this it is more convinient to measure so-called round-trip delay or the time difference between sending a packet and receiving an acknowledgement that the packet has reached its destination without any information loss. Both time instants are easily observable within the transport station. If the measurements are carried out by a single monitoring system, the delay can be evaluated from the difference between the time-stamps of the measurement packet or especially interpretable data packet.

Reliability interpreted as the error rate and the retransmission rate can be evaluated by observation of the error control function. The evaluation of the reliability criteria is straightforward.

Efficiency can be evaluated from the knowledge of retransmissions number and the bit format of the transport-data unit. The number of retransmissions is observable through the error control function.

Availability can be evaluated by memorizing failures to send the transport-data units due to unavailability of the transport connection.

The above remarks on the practical evaluation of the proposed quality of service criteria can serve as guidelines for design of the measurement tools for the transport station monitoring module. It should be noted that if, in the network each system /each layer/ has a monitoring module of its own, the utilization of the measurement packets or especially interpretable data packets is not required. The above packets will be needed for other purpose like aggregating the mass measurement data at the off-line analysis center.

IX. MEASUREMENT OF OTHER TRANSPORT LAYER PARAMETERS

The measurement program for evaluation of the proposed quality of service criteria can be extended into three directions:

- for more complete determination of the quality of the transport services,
- for selection of mechanisms performing the transport layer functions,
- for evaluation of other layer characteristics different from the quality of service.

In the first case, measurements can be applied to the other services of the layer and their quality can be evaluated. Once they are included in the network, the transaction - oriented services and broadeast - oriented services can be objects of such extended measurements.

In the second case, the values of appropriate criteria would constitute an important information for the transport protocol designer supporting or refuting his design solutions or just suggesting some modifications. Thus, such an extended measurement program can be identified as a tool for practical design verification. We particularly suggest the evaluation of the flow control mechanism, due to its impact on majority of the proposed transport connection quality of service criteria. Besides, a badly designed flow control mechanism can cause the congestion phenomenon in the network resulting in a partial or total packet immobility. Another matter of interest from the designer viewpoint is the quality of service of the error recovery mechanism in the transport layer /for error notifications from the network control layer/ and of the class of service negotiation mechanism. Hence the above extentions can be identified as a tool for verification of the transport protocol design.

In the third case, some statistics of the transport layer behavior can be evaluated by the off-line analysis tool. The

objective of such measurements is gathering data for statistical reduction and the resultant statistics can reflect:

- the utilization of the transport layer resources,
- the occurence of events within the layer.

Memory buffers and processing time required for the operation of the transport station are the most common transport layer resources and their observability is straightforward.

Among multitude of events occuring within the transport layer, of special importance are the ones allowing for evaluation of:

- time characteristics of establishment, maintainance and
- termination of the transport connections,
- traffic characteristics of the transport connections.

Before developping the extended measurement program, it should be noted that excessive measurements can cause severe degradations of the network services and that, due to the itroduced artifact, the measurements themselves can be corrupted.

X. FINAL REMARKS

In compliance with guidelines suggested for early stages of the measurement planning [10], we have presented four basic components of the future experiment:

/i/ object of measurements - the transport layer within the frame of the Open Systems Architecture,

/ii/ objectives of measurements - evaluation of the quality of services provided by the transport layer,

/iii/ quality of service criteria - the proposed set of the quality of service criteria,

/iv/ measurement tools - the presented methods applicable for carrying out measurement experiments.

The measurement program suggested in our paper can only be presented in an outline, mainly due to early stages of the network development and hence to the absence of the transport protocol.

We have made our considerations deliberately general, so that the measurement program can be further developped for any version of the transport protocol. As the designers of the inter-university network have adopted the ISO 7-layered view of the architecture, we have found the transport layer within the frame of the architecture particularly suitable for our purpose. In the course of the measurement software development, the measurement program will probably have to be modified and once the network is operational, the program should be up-graded to accomodate yet another monitoring services, as suggested in the Chapter IX. It goes without saying that the software should be made as versatile and expandable as possible to facilitate any minor or major modifications.

The next remark concerns the possibility of controlling the transport layer quality of service by means of a "short" feedback loop. This possibility is implied in the ISO document [1] and it is undoubtedly an attractive proposal for a local improvement of the network behavior. However, we suggest that the control mechanism be included in the network at some further stages and in the meanwhile the measurement data can be sent to the human analyst for off-line analyses. Our suggestion is more obvious when it is realized that at the beginning, 1:1 mapping of connections will only be performed, resulting in fairly equal qualities of services for every transport connection. Once the differences between them begin to appear, /by introduction of the multiplexing rules/, the quality of service control mechanism should be included in the network. To design and to implement this mechanism is a challenging task for the network designers.

Finally, it is necessary to point out that this research is a preliminary stage of a broad measurement program of the multi-layered network in which there are many unsolved problems.

REFERENCES

[1] "Reference model of Open Systems Architecture", version 3 as of November 1978, ISO/TC97/SC16N117.

[2] J.A. Barchański, M. Bazewicz, "A preliminary project of an inter-university computer network", November 1978 /in Polish/.

[3] E. Bilski, L. Budzianowski, T. Muehleisen, E. Sorokin, J. Wietrzych, W. Żabnieński, "A FEP for the inter-university network - design aspects", report spr11, October 1979 /in Polish/.

[4] CCITT, "Recommendation X.25: Interface between data terminal equipment /DTE/ and data circuit-terminating equipment /DCE/ for terminals operating in the packet mode on public data networks", Orange Book, vol. VIII.2, Sixth Plenary Assembly, Int. Telecommunications Union, Geneva, Switzerland, 1977.

[5] G.D. Cole, "Computer network measurements: techniques and experiments", Engineering Report No. UCLA-ENG-7165, University of California, Los Angeles, Calif., 1971.

[6] C.J. Bennet, A.J. Hinchley, "Measurements of the transmission control protocol", in Proc. Workshop on Computer Network Protocols /Liege, Belgium/, February 1978.

[7] J.L. Grange, P. Mussard, "Performance measurements of line control protocols in the Cigale network", in Proc. Workshop on Computer Network Protocols /Liege, Belgium/, February 1978.

[8] L. Kleinrock, W.E. Naylor, "On measured behavior of the ARPA network", in AFIPS Conf. Proc. 1974, National Computer Conference, vol. 43, pp. 767-780, 1974.

[9] L. Kleinrock, H. Opderback, "Throughput in the ARPANET - protocols and measurements", in Proc. 4th Data Communication Symp. /Quebec City, Canada/, October 1975.

[10] F.A. Tobagi, M. Gerla et al., "Modelling and measurements techniques in packet communication networks", in Proc. of the IEEE, vol. 66, pp. 1423-1447, November 1978.

[11] J.A. Barchański, "Evaluation of acknowledgement strategies for packet-switched computer networks", in Proc. Remote Data Transmission '79, Karlovy Vary, Czechoslovakia, September 1979.

Raport złożono w Redakcji
w styczniu 1980 r.

dr inż. J.A. Barchański
mgr inż. T. Muehleisen

ПЕРЕДАЧА СООБЩЕНИЙ В СЕТЯХ ЭВМ С ДИНАМИЧЕСКИМ ПРИОРИТЕТОМ

Секер Иштван, Венгрия

Сеть ЭВМ образуется соединением территориально распределенных вычислительных мощностей и устройств пользователей с помощью сети передачи данных.

Сеть передачи данных содержит все те технические и программные средства, которые обеспечивают доставку сообщений, имеющих стандартный формат /в конкретном случае формат в рамках соглашения, наперед определенного для данной сети/, от узла отправки к узлу адресата.

Сеть передачи данных формально можно описать как мультиграф, где описываются узлы и соединения между ними. Соединения реализуются дискретными каналами передачи данных, а узлы средствами коммутации.

В сети передачи данных узлы коммутации и каналы характеризуются различными параметрами, зависящими от их работы /напр. узел: дисциплина обслуживания и маршрутизации сообщений; каналы: скорость передачи данных, дуплексный, полудуплексный режимы работы/. На основе этих параметров можно определить параметры сети /напр. среднее время задержки сообщений в сети или в конкретном пути сети/.

Задача сети передачи данных состоит в обеспечении потоков информации между узлами стока и истока /для определенных потоков/ через сеть с заданными качествами времени, достоверности и надежности.

В сети передачи данных можно образовать пути между парами узлов x_i и x_j, в общем случае несколько путей для данных пар. Это ставит задачу определения метода или стратегии выбора путей потоков. На основе исследования и практики /лит. 1,2./:

а/ в целях минимизации времени задержки информации в сети передачи данных, целесообразно передавать информацию по фиксированным путям между источниками и адресатами /фиксированная маршрутизация/.

б/ в целях увеличения надежности целесообразно применять альтернативные пути между источниками и адресатами /случайная, альтернативная, адаптивная маршрутизации/.

Пункты а/ и б/ противоречивы: надежность увеличивается за счет увеличения времени задержки и наоборот. Учитывая требования к качеству работы сети передачи данных можно выбрать подходящий метод маршрутизации.

Если надежность составляющих элементов сети высокая, то работа с фиксированной маршрутизацией может оказаться целесообразной, но при этом временная перегрузка каналов некоторых регионов значительно уменьшает качество передачи данных между определенными парами узлов: уменьшается используемость определенных путей. Фиксированной маршрутизацией нет выхода из этого положения. Это приведет к уменьшению результирующей надежности всей сети. В случае большой сети временная перегрузка отдельных регионов сети не влияет на среднее значение времени задержки данных для всей сети, потому-что рядом с перегруженными, есть и недогруженные регионы. Однако расброс времени задержки увеличивается при перегрузке регионов.

Если надежность элементов сети невысокая, то:

а/ в каждом узле выбирается далбнейшее направление сообщения случайно из работоспособных выходящих линий по случайному методу образования путей. При этом не надо предпринимать специальные меры учета повреждений элементов сети и учитывать адрес адресата сообщения. Если не нарушается связность сети, то сообщения обязательно дойдут до адресата. При этом средняя длина пути может оказаться во много раз больше средней длины пути при фиксированной маршрутизации сообщений.

б/ сообщения передаются по фиксированным путям до тех пор, пока сеть находится в определенном состоянии. При изменении состояния сети пути тоже изменяются, приводятся в соответствие с состоянием сети. Степень и частота изменения путей зависит от расстояний между

состояниями и определяет, насколько отошли от фиксированной маршрутизации к альтернативной.

Предлагается метод обслуживания сообщений, с помощью которого:

а/ можно увеличить качество работы сети передачи данных, работающей с фиксированной маршрутизацией, путем компенсации перегрузок регионов за счет недогруженных регионов сети,

б/ можно уменьшить частоту изменения состояний сети при альтернативной маршрутизации, исключая из условий изменения состояний перегрузку регионов,

в/ можно произвести анализ работы сети для усовершенствования ее работы.

Метод базируется на приоритетном обслуживании, при котором приоритет имеет динамический характер: определяется актуальным значением срочности сообщения.

Описание метода приоритетного обслуживания:

Определим сообщение с точки зрения передачи данных:

Сообщение - это существенная для пользователя и/или устройства обработки данных информация, преобразованная /редактированная/ в формат, поддающийся передаче через сеть передачи данных. Сообщение есть субъект обслуживания сетью.

Дадим другое определение сети передачи данных:

Сеть передачи данных есть совокупность совместно работающих обслуживающих приборов - каналов передачи данных и средств организации обслуживаний - узлов коммутации.

Рассмотрим как происходит работа сети ЭВМ, состоящей из сети передачи данных, мощных ЭВМ и устройств пользователей.

Пользователь отправляет сообщение к ЭВМ большой мощности, заявку, в которой определяет нужную службу для решения своей проблемы /вычисление, получение информации, регистрация событий.../. Пользователь ждет ответного сообщения о выполнении и о результате выполнения службы. Ответное сообщение должно прибыть к пользова-

телю за определенное время: $T_{крит}$ от момента отправления заявки. Это время называется критическим временем ответа. За это время происходят следующие события:

а/ доставка заявки от пользователя к ЭВМ,

б/ обработка заявки, выполнение нужной службы,

в/ доставка ответного сообщения от ЭВМ к пользователю.

Для того, чтобы ответ пришел за $T_{крит}$ от момента отправки заявки, должно выполняться следующее неравенство:

$$T_{крит} \geqslant T_{q\,заявка} + T_{обр} + T_{q\,ответ} \qquad / I /$$

где $T_{q\,заявка}$ – время события по п. а/,

$T_{обр}$ – время события по п. б/,

$T_{q\,ответ}$ – время события по п. в/.

Если рассмотрим крайний случай неравенства / I /, то:

$$T_{крит} = T_{q\,заявка\,крит} + T_{обр\,крит} + T_{q\,ответ\,крит} \qquad / 2 /$$

При телеобработке данных в реальном масштабе времени правая и левая части неравенства / I / имеют одинаковый порядок и приблизительно одинаковы. При пакетной обработке данных значение левой части может быть существенно больше правой части.

Задача сети ЭВМ при работе в реальном масштабе времени состоит в соблюдении неравенства / I / для возможно большого количества отправленных заявок, т.е. сеть ЭВМ должна работать с малыми потерями.

В случае пакетной обработки данных задача состоит в уменьшении среднего значения времени ответа для большого количества заявок, что равнозначно увеличению пропускной способности сети ЭВМ. При этом время ответа может иметь большой разброс.

Рассмотрим как можно обеспечить неравенство / I / в случае работы в реальном масштабе времени.

Легко видно, что при выполнении следующей системы не-

равенств:

$$\overline{T}_{крит} = \overline{T}_{q\,заявка\,крит} + \overline{T}_{собр\,крит} + \overline{T}_{ответ\,крит}$$

$$\overline{T}_{q\,заявка\,крит} \geqslant \overline{T}_{q\,заявка}$$

$$\overline{T}_{собр\,крит} \geqslant \overline{T}_{собр} \qquad / 3 /$$

$$\overline{T}_{q\,ответ\,крит} \geqslant \overline{T}_{q\,ответ}$$

неравенство / I / тоже выполняется.

На основе равенства системы неравенств / 3 / определяем левые части неравенств, и при работе сети ЭВМ соблюдаем выполнение неравенств.

Транспортировка сообщений происходит в сети передачи данных, таким образом выполнение двух неравенств:

$$\overline{T}_{q\,заявка\,крит} \geqslant \overline{T}_{q\,заявка}$$

$$/ 4 /$$

$$\overline{T}_{q\,ответ\,крит} \geqslant \overline{T}_{q\,ответ}$$

налогается на сеть передачи данных.

Время ответа содержит два времени доставки сообщений. Для сети передачи данных безразлично какое сообщение передается: заявка или ответ, т.к. не знает /и не может знать/ о содержимом сообщений. Поэтому в дальнейшем индексы "$_{заявка}$" и "$_{ответ}$" опускаются.

Таким образом можно сформулировать задачу сети передачи данных, работающей в реальном масштабе времени: доставка сообщений от места отправки до места назначения за время $\overline{T}_{q\,крит}$.

Пусть сообщение снабжается счетчиком, который при отправке /момент: $t_{отпр}$ / по сети получит значение $\overline{T}_{q\,крит}$. Значение счетчика во время транспортировки сообщения /нахождение в сети передачи данных/:

$$\overline{T}_{q\,ост} = \overline{T}_{q\,крит} - (t - t_{отпр}) \qquad / 5 /$$

где $T_{q\,ост}$ – остаточное время транспортировки,

$t_{отпр}$ – время отправки сообщения.

Этот счетчик назовем счетчиком остаточного времени: его значение все время показывает, сколько времени может сообщение находиться в сети передачи данных до окончания его доставки к месту назначения.

По значениям счетчиков доставленных сообщений можно произвести анализ работы сети передачи данных:

I/ Имеется ли соответствие между требуемым временем доставки и действительным значением времени доставки сообщений в сети?

Если среднее от конечных значений счетчиков остаточного времени сообщений намного больше среднего времени доставки сообщений сетью:

$$\frac{1}{n}\sum_{i=1}^{n} T_{q\,ост\,кон\,i} \gg \overline{T}_{q\,сети} \qquad /6/$$

где n – количество доставленных сообщений,

$\overline{T}_{q\,сети}$ – среднее время доставки сообщений сетью,

$T_{q\,ост\,кон\,i}$ – значение счетчика остаточного времени i-го сообщения после его доставки,

то сеть рассчитана на большую нагрузку действительной нагрузки.

2/ Каковы потери сообщений относительно всей сети и ее отдельных путей?

Рассмотрим сумму:

$$A = \sum_{i=1}^{n} a_i$$

где $a_i = 1$ если $T_{q\,ост\,кон\,i} \geq 0$,

$a_i = 0$ если $T_{q\,ост\,кон\,i} < 0$.

Вероятность своевременной доставки сообщений относительно сети:

$$P_{сети\,дост} = \frac{\sum_{i=1}^{n} a_i}{n} \qquad /7/$$

Вероятность своевременной доставки сообщений относи-

пути между двумя узлами x_j и x_k :

$$p_{jk\,дост} = \frac{\sum_{e=1}^{m_{jk}} a_e}{m_{jk}} \qquad / 8 /$$

где m_{jk} – количество сообщений, отправленных от узла x_j и доставленных в адресованный узел x_k .

Взвешенное среднее значение вероятностей доставки относительно путей сети дает вероятность доставки относительно всей сети:

$$P_{сети\,дост} = \sum_j \sum_k p_{jk\,дост} \frac{m_{jk}}{n} \qquad / 9 /$$

где $\sum_j \sum_k m_{jk} = n$.

Вероятность потери сообщений сети и путей соответственно:

$$P_{сети\,пот} = 1 - P_{сети\,дост}$$

$$p_{jk\,пот} = 1 - p_{jk\,дост} \qquad / 10 /$$

Из / 7,8,9 и 10 / следует:

$$P_{сети\,пот} = \sum_j \sum_k p_{jk\,пот} \cdot \frac{m_{jk}}{n} \qquad / 11 /$$

3/ Какие регионы являются узкими месиами сети передачи данных?

Если в результате анализа конечных значений счетчиков остаточного времени оказывается, что для некоторых путей сети вероятность потери значительно больше вероятности потери относительно всей сети:

$$p_{jk\,пот} \gg P_{сети\,пот} \qquad / 12 /$$

и эти пути имеют общность в том, что все проходят через один и тот же регион сети, т.е.:

$$Region = \bigcap_{j,k} P_{jk} \neq \emptyset$$

где P_{jk} – пути, для которых выполняется / I2 /, тогда подмножество, полученное пересечением этих путей:

$$Region = \bigcap_{j,k} P_{jk}$$

есть регион сети, являющийся узким ее местом. Это значит, что

- внутри этого региона установить новые каналы передачи данных,
- необходимо установить новые каналы около этого региона для обеспечения обхода региона частью потока данных.

Рассмотрим сеть передачи данных, каксистему обслуживания, куда поступают сообщения, требуют и получают транспортное обслуживание, после которого покидают ее.

Обслуживание сообщения сетью передачи данных – это есть транспортировка сообщения по пути, состоящему из одного или нескольких участков – каналов передачи данных. Значит обслуживание сообщения сетью есть суммарное обслуживание сообщения составляющими пути каналами.

Характер поступления сообщений считаем пуассоновским.

Сообщение, проходя через сеть, поступает в несколько узлов сети где:

а/ или ждет и получает дальнейшее обслуживание одним из выходящих из данного узла каналов, если узел промежуточный по пути сообщения,

б/ или покидает сеть передачи данных, если узел конечный по пути.

Каналы передачи данных являются обслуживающими приборами, которые частично обслуживают сообщения.

Основные задачи сети передачи данных состоят:

а/ в составлении путей сообщений по заданному алгоритму маршрутизации. Это для данного узла означает, что узел решает вопрос о том, что сообщение с заданным местом назначения /адресом адресата/ какой выходящей линией будет обслуживаться,

б/ в организации дисциплины обслуживания сообщений: в составлении очередей перед каналами передачи данных.

Побочные задачи сети:

а/ создание условий для выполнения основных задач: сбор статистических данных, работа со служебными сообщениями /выработка, передача, прием, обработка/,

б/ увеличение работоспособности сети, обеспечение удобной работы пользователей с вычислительными мощностями через сеть передачи данных /ограничение количества поступающих в сеть сообщений – управление потоками данных, передача сообщений в определенный момент времени по просьбе пользователя.../.

Известна теорема о независимости обслуживающей работы каналов передачи данных в сети передачи данных /лит. I/. По этой теореме в сети передачи данных каналы обслуживают независимые друг от друга пуассоновские потоки сообщений.

Из теории массового обслуживания известна формула Хинчина – Поллачека, которая определяет связь между средним значением пребывания объекта обслуживания в системе обслуживания в зависимости от занятости обслуживающего прибора:

$$t_q = t_s \cdot H(\varrho)$$

где t_s – среднее время обслуживания,

ϱ – занятость, нагруженность обслуживающего прибора,

$t_q = t_w + t_s$ – время пребывания объекта обслуживания в системе обслуживания – системное время,

t_w – время ожидания на обслуживание,

H – формула Хинчина – Поллачека.

В случае пуассоновского характера поступления заявок на обслуживание и экспоненциального распределения времени обслуживания:

$$H(\varrho) = \frac{1}{1-\varrho}$$

Если не учитываем регулярного поступления заявок, то $H(\varrho)$ монотонно возрастающая функция от ϱ, и:

$$\lim_{\varrho \to 0} H(\varrho) = 1$$

$$\lim_{\varrho \to 1} H(\varrho) = \infty$$

ϱ определяется как отношение времени занятости обслуживающего прибора ко времни периода работы:

$$\varrho = \frac{\sum^{T} t_{зан}}{\sum^{T} t_{зан} + \sum^{T} t_{своб}} = \frac{1}{T} \sum^{T} t_{зан}$$

где $\sum^{T} t_{зан}$ – время действительной работы за T,

$T = \sum^{T} t_{зан} + \sum^{T} t_{своб}$ – время периода работы,

$\sum^{T} t_{своб}$ – время свободного состояния за T.

В случае передачи данных:

$$\varrho = \frac{\varphi}{V} \qquad / 13 /$$

где φ – поток информации через канал,

V – скорость передачи данных.

Поток информации описывается выражением:

$$\varphi = \frac{\lambda}{\mu} \qquad / 14 /$$

где λ – интенсивность поступления сообщений,

μ – средняя длина сообщений.

Поставляя / 14 / в / 13 / :

$$\rho = \frac{\lambda}{\mu V} \qquad / \text{I5} /$$

Если сообщение проходит через сеть передачи данных, то получает обслуживание, качество которого можно характеризовать соотношением:

$$H(\bar{\rho}_{пути}) = \frac{\sum\limits_{i \in каналы\ пути} t_{q_i}}{\sum\limits_{i \in каналы\ пути} t_{s_i}} \qquad / \text{I6} /$$

где i – индексы каналов.

Чем меньше это соотношение, тем лучше качество обслуживания. Область значения дроби / I6 /: $[1, \infty)$. По / I6 / можно оценить результирующую нагруженность пути сообщения в сети:

$$\bar{\rho}_{пути} = H^{-1}\left(\frac{\sum\limits_i t_{q_i}}{\sum\limits_i t_{s_i}}\right) = H^{-1}\left(\frac{\sum\limits_i H(\rho_i) \cdot t_{s_i}}{\sum\limits_i t_{s_i}}\right) \qquad / \text{I7} /$$

Если нагруженность всех каналов пути одинакова:

$$\rho_i = \rho_j \qquad i, j \in Каналы\ пути$$

то из / I7 / следует:

$$\bar{\rho}_{пути} = \rho_i \qquad / \text{I8} /$$

По дисциплине обслуживания сообщений "первым приходишь первым обслуживаешься" сообщения обслуживаются одинаковым качеством. Это значит, что соотношение / I6 / и нагруженность пути, которую путь показывает сообщениям, проходящим по одному и тому же пути, одинакова для всех сообщений:

$$\frac{\sum\limits_i t^{\ell}_{q_i}}{\sum\limits_i t^{\ell}_{s_i}} = \frac{\sum\limits_i t^{r}_{q_i}}{\sum\limits_i t^{r}_{s_i}}$$

и следовательно:

$$\bar{S}^{l}_{пути} = S^{r}_{пути} \qquad / 19 /$$

где l, r – индексы сообщений,

i – индексы каналов пути.

Но это не соответствует действительному требованию на качество обслуживания сообщений.

Сообщение, поступающее в сеть передачи данных, может пребывать в сети на время $\bar{T}_{крит}$. Если время обслуживания сообщения сетью:

$$\bar{T}_S = \sum_{i \in \text{каналы пути}} t_{Si} \qquad / 20 /$$

то требование сообщения на качество обслуживания выражается соотношением:

$$\frac{\bar{T}_{q\,крит}}{\bar{T}_S} \qquad / 21 /$$

Качество обслуживания должно быть не хуже требуемого качества:

$$\frac{\bar{T}_{q\,крит}}{\bar{T}_S} \geqslant \frac{\sum_{i} t_{qi}}{\sum_{i} t_{Si}} \qquad / 22 /$$

Во время пребывания сообщения в сети передачи данных, оно может еще пребывать в сети на время $\bar{T}_{q\,ост}$, а за это время необходимо получать обслуживание с общим временем:

$$\bar{T}_{S\,ост} = \sum_{j \in \text{остальные каналы пути}} t_j \qquad / 23 /$$

чтобы оно было вовремя доставлено на место назначения. Это время назавем остаточным временем обслуживания сообщения. Требуемое от остальной части пути качество обслуживания определяется как соотношение:

$$\frac{\overline{T}_{q\,ост}}{\overline{T}_{s\,ост}}$$

такое, что:

$$\frac{\overline{T}_{q\,ост}}{\overline{T}_{s\,ост}} \geqslant \frac{\sum_{d} t_{qd}}{\sum_{j} t_{sj}} \qquad / 24 /$$

где j – индексы остальных каналов пути.
Неравенство / 24 / включает в себя неравенство / 22 / на основе выражения:

$$\overline{T}_{q\,ост} = \overline{T}_{q\,крит} - (t - t_{отпр})$$

$$\frac{\overline{T}_{q\,ост}(t = t_{отпр})}{\overline{T}_{s\,ост}(t = t_{отпр})} = \frac{\overline{T}_{q\,крит}}{\overline{T}_{s}}$$

и

$$\frac{\sum_{j} t_{qj}}{\sum_{j} t_{sj}} = \frac{\sum_{i} t_{qi}}{\sum_{i} t_{si}}$$

где i – индексы каналов пути,
j – индексы остальных каналов пути при $t = t_{отпр}$.

Рассмотрим как используется счетчик остаточного времени для приоритетного обслуживания: для выбора обслуживаемого сообщения из ожидающих.

При освобождении канала сообщение выберем из ожидающх обслуживания этой линией на основе значений соотношения

$$K(t) = \frac{\overline{T}_{q\,ост}(t)}{\overline{T}_{s\,ост}(t)} \qquad / 25 /$$

сообщений. Это соотношение обозначим K и назавем параметром приоритета сообщения.

Параметр приоритета сообщения:

I/ Учитывает срочность сообщения, тем самым показы-

вает, каким качеством должна остальная часть пути обслужить сообщение для своевременной доставки.
Если сообщение имеет длину, равняющуюся средней длине сообщений: $\mu^j = \bar{\mu}$

$$\rho^j_{ост} \leq H^{-1}(K_j(t)) \qquad / 26 /$$

$$\rho^j_{ост\,доп} = H^{-1}(K_j(t)) \qquad / 26.а /$$

где $\rho^j_{ост}$ – нужное значение нагруженности остальной части пути для сообщения j ,
$\rho^j_{ост\,доп}$ – допустимое значение нагруженности остальной части пути для сообщения j .
Если длина сообщения отличается от средней длины:

$$\mu^j \neq \bar{\mu}$$

$$\rho^j_{ост} \leq H^{-1}\left(\frac{\mu^j}{\bar{\mu}}(K_j(t)-1)+1\right) \qquad / 27 /$$

$$\rho^j_{ост\,доп} = H^{-1}\left(\frac{\mu^j}{\bar{\mu}}(K_j(t)-1)+1\right) \qquad / 27.а /$$

Вывод неравенств / 26 и 27 /:
Среднее время пребывания сообщения средней длины в системе:

$$\bar{\bar{T}}_q(\bar{\mu}) = H(\rho)\cdot \bar{T}_s(\bar{\mu})$$

или

$$\bar{\bar{T}}_q(\bar{\mu}) = \bar{\bar{T}}_\omega(\bar{\mu}) + \bar{T}_s(\bar{\mu})$$

где $\bar{\bar{T}}_\omega(\bar{\mu})$ – время ожидания на обслуживание в системе.
В случае передачи данных:

$$\bar{\bar{T}}_\omega(\bar{\mu}) = \sum_{i\,\text{каналы пути}} t_{\omega_i}(\bar{\mu})$$

Время ожидания на обслуживание не зависит от длины сообщения:

$$\overline{\overline{T}}_w(\bar{\mu}) = \overline{\overline{T}}_w(\mu^{-\delta})$$

следовательно:

$$\overline{\overline{T}}_w(\mu^{-\delta}) = \overline{\overline{T}}_q(\mu^{-\delta}) - \overline{T}_s(\bar{\mu}) = \left(H(\varsigma) - 1\right) \overline{T}_s(\bar{\mu}) .$$

С учетом того, что

$$\overline{T}_s(\bar{\mu}) = \overline{T}_s(\mu^{-\delta}) \frac{\bar{\mu}}{\mu^{-\delta}}$$

получим:

$$\overline{\overline{T}}_w(\mu^{-\delta}) = \left(H(\varsigma) - 1\right) T_s(\mu^{-\delta}) \frac{\bar{\mu}}{\mu^{-\delta}} .$$

Отсюда выразим $H(\varsigma)$:
Учитывая, что

$$\overline{T}_w(\bar{\mu}) = \overline{T}_q(\mu^{-\delta}) - \overline{T}_s(\mu^{-\delta}) \quad :$$

$$H(\varsigma) = \frac{\mu^{-\delta}}{\bar{\mu}} \left(\frac{\overline{\overline{T}}_q(\mu^{-\delta})}{\overline{T}_s(\mu^{-\delta})} - 1 \right) + 1 .$$

Так как $H(\varsigma)$ монотонно возрастающая функция:

$$\varsigma = H^{-1}\left(\frac{\mu^{-\delta}}{\bar{\mu}} \left(\frac{\overline{\overline{T}}_q(\mu^{-\delta})}{\overline{T}_s(\mu^{-\delta})} - 1 \right) + 1 \right) .$$

Учитывая, что

$$\overline{T}_{q\,oct}(\mu^{-\delta}) \geqslant \overline{T}_q(\mu^{-\delta})$$

и

$$K_j = \frac{\overline{T}_{q\,oct}(\mu^{-\delta})}{\overline{T}_s(\mu^{-\delta})} \geqslant \frac{\overline{T}_q(\mu^{-\delta})}{\overline{T}_s(\mu^{-\delta})}$$

а также, что H^{-1} монотонно возрастающая функйия:

$$\varsigma \leqslant H^{-1}\left(\frac{\mu^{-\delta}}{\bar{\mu}} (K_j - 1) + 1 \right)$$

Если $\tau^j = \bar{\tau}$, то:

$$\rho^j_{ост} \leq H^{-1}(K_j(t))$$

2/ Учитывает функциональное расстояние от места назначения.

Параметр приоритета выражается формулой:

$$K_j(t) = \frac{\bar{T}^j_{q\,ост}(t)}{\bar{T}^j_{s\,ост}(t)} = \frac{\bar{T}^j_{q\,крит} - (t - t^j_{стр})}{\bar{T}^j_{s\,ост}(t)} = \qquad / 28 /$$

$$= \frac{\bar{T}^j_{q\,ост}(t_i)}{\bar{T}^j_{s\,ост}(t)} - \frac{t - t^j_i}{\bar{T}^j_{s\,ост}(t)}$$

где j – индекс сообщения,

t^j_i – время поступления j -го сообщения в i -ый узел пути,

$\bar{T}^j_{s\,ост}(t_i)$ – остаточное время обслуживания j -го сообщения, находящего в i -ом узле пути:

$$\bar{T}^j_{s\,ост}(t_i) = \bar{T}^j_{s\,ост}(t) \quad при\ t_i \leq t < t_{i+1}$$

Значение $\bar{T}^j_{s\,ост}(t)$ тем меньше, чем ближе находится сообщение к месту назначения, т.е. чем меньше обслуживание /выраженное во времени передач/ надо для его доставки. Значение $K(t)$ убывает быстрее у сообщений, $\bar{T}^j_{s\,ост}(t)$ которых меньше:

$$\frac{d\,K_j(t)}{dt} = -\frac{1}{\bar{T}^j_{s\,ост}(t)}$$

Сообщения, требующие меньшего обслуживания от сети передачи данных для завершения их доставки, быстрее подбираются вперед в очереди. Это естественно необходимо, потому-что в случае какой-либо задержки они имеют меньше шанса отработать эту задержку.

3/ $K_j(t)$ не имеет смысла на месте назначения. На

месте назначения $T^{j}_{s\,ост}(t) = 0$, нет нужд в значении K, т.к. не надо выбирать для обслуживания /передачи/ сообщение, прибывшее на место назначения.

Рассмотрим как образуется параметр K для сообщений и как обращаемся с сообщениями в узле коммутации.

Сообщение с номером j отправляется из узла k в узел ℓ, узлы пути $P_{k\ell}$ получают новую нумерацию. Номером узла будет его порядковый номер по проходу пути. Если путь $P_{k\ell}$ состоит из n каналов, то узел k получит номер 0 и узел ℓ номер n. Сообщение имеет счетчик остаточного времени, начальное значение которого при отправке /при ставке в очередь передачи в 0 -ом узле пути/ устанавливается:

$$T^{j}_{q\,ост}(t = t_0) = T^{j}_{q\,крит} \qquad / 29 /$$

а/ сообщение поступает в какой-то промежуточный узел /i -ый/ пути. При поступлений $T^{j}_{q\,ост\,i}$ будет иметь значение:

$$T^{j}_{q\,ост\,i} = T^{j}_{q\,ост}(t = t^{j}_{i}) \qquad / 30 /$$

где t^{j}_{i} – время поступления j -го сообщения в i -ый узел.

б/ программа анализатор вынимает из сообщения адрес адресата и значение счетчика остаточного времени.

в/ на основе адреса адресата по таблице маршрутизации определяются:

- направление передачи сообщения /выходящий канал/,
- время, необходимое для передачи сообщения средней длины по остальной части пути /остаточное время обслуживания сообщения средней длины/: $T_{s\,ост\,i}(\bar{\mu})$.

г/ вычисляется остаточное время обслуживания j -го сообщения:

$$T^{j}_{s\,ост\,i}(\bar{\mu})\,\frac{\mu^{j}}{\bar{\mu}} \qquad / 31 /$$

где $\bar{\mu}$ – средняя длина сообщений,

μ^j – длина j -го сообщения.

д/ вычисляется начальное значение $K_j(t=t_i)$

$$K_j(t=t_i) = \frac{T^j_{q\,ост}(t=t_i)}{T^j_{s\,ост_i}} \qquad / 32 /$$

и производится анализ $K_j(t=t_i)$. Для анализа примим, что если передача некоторого сообщения началась, то передачу нельзя прерывать для обслуживания более срочного сообщения.

– если

$$K_j(t=t_i) \geq 1 + \bar{\varrho}_{сети} \frac{\mu^j}{\bar{\mu}} \qquad / 33 /$$

где $\bar{\varrho}_{сети}$ – ожидаемое значение нагруженности сети:

– определяется анализом служебных сообщений центральным управлением сетью,

– устанавливается при проектировании сети,

– получает для анализа максимальное значение /наихудший случай/: $\bar{\varrho}_{сети} = 1$,

то сообщение имеет шансы на своевременную доставку.

– если

$$K_j(t=t_i) < 1 + \bar{\varrho}_{сети} \frac{\mu^j}{\bar{\mu}} ,$$

то сообщение не имеет шансов на своевременную доставку: необходимо предпринимать специальные меры:

– несмотря на это передать сообщение,

– стереть сообщение.

Обоснование критерия / 33 /:

Если сообщение имеет максимальный приоритет в каждом узле, то необходимое время пребывания в сети есть остаточное время обслуживания и время ожиданий окончаний обслуживаний уже обслуживаемых сообщений:

$$\overline{T}^{\delta}_{q\,ост_i}(\mu^{-\delta}) \geq \overline{T}^{\delta}_{s\,ост_i}(\mu^{-\delta}) + \bar{\xi}_{сети}\,\overline{T}_{s\,ост_i}(\bar{\mu})$$

и так как:

$$\overline{T}_{s\,ост_i}(\bar{\mu}) = \overline{T}^{\delta}_{s\,ост_i}(\mu^{-\delta})\,\frac{\bar{\mu}}{\mu^{-\delta}}$$

получим:

$$\frac{\overline{T}^{\delta}_{q\,ост_i}(\mu^{-\delta})}{\overline{T}^{\delta}_{s\,ост_i}(\mu^{-\delta})} \geq 1 + \bar{\xi}_{сети}\frac{\bar{\mu}}{\mu^{-\delta}}$$

что равносильно:

$$K_{\delta}(t=t_i) \geq 1 + \bar{\xi}\,\frac{\bar{\mu}}{\mu^{-\delta}}$$

е/сообщение ставится в динамическую очередь перед передачей в определенном направлении. Динамическая очередь означает, что порядок элементов в очереди не устанавливается при поступления нового элемента в очередь. Порядок элементов в динамической очереди является функцией времени так, чтобы элементы были упорядочены по некоторой величине, зависящей от времени. Эту величину назавем коэффициентом приоритета.
Коэффициентом приоритета для нашего случая выберем выражение:

$$P_r(t) = \left(K_{\delta}(t) - 1\right)\cdot\frac{\mu^{-\delta}}{\bar{\mu}} \qquad / 34 /$$

Нет необходимости в $\bar{\mu}$ в знаменателе дроби, но для рассмотрения целесообразно оставить. Для реализации динамической очереди можно принимать:

$$P_r(t) = \left(K_{\delta}(t) - 1\right)\mu^{-\delta} \qquad / 35 /$$

Обоснование выбора коэффициента приоритета сообщения:

То сообщение обслуживается первым, которому необходимо оказать обслуживание наивысшим качеством, т.е. для которого $\varrho^{j}_{\text{ост доп}}$ / 27.а / имеет минимальное значение:

$$\varrho^{j}_{\text{ост доп}} = H^{-1}\left(\frac{\mu^{-j}}{\bar{\mu}}(K_j(t)-1)+1\right)$$

H^{-1} монотонно возрастающая функция, минимальное значение имеет для того сообщения, которое имеет минимальное значение коэффициента приоритета.
Рассмотрим $P_r(t)$ по / 34 /:

$$P_r(t)=(K_j(t)-1)\frac{\mu^{-j}}{\bar{\mu}}=\frac{\bar{T}^{j}_{q\,\text{ост}_i}(t)-\bar{T}^{j}_{s\,\text{ост}_i}(\mu^{-j})}{\bar{T}^{j}_{s\,\text{ост}_i}(\bar{\mu})} \qquad / 36 /$$

Это значит, что коэффициент приоритета учитывает время, которое сообщение может затратить на ожидание дальнейших обслуживаний, и функциональное расстояние сообщения средней длины от места назначения.

Рассмотрим пример:
Отправляем два сообщения r и s одновременно по одному пути P_{il}. Оба сообщения получают одинаковые критические времена:

$$\bar{T}^{r}_{q\,\text{крит}} = \bar{T}^{s}_{q\,\text{крит}}$$

Длины сообщений μ^{-r} и μ^{-s} такие, что:

$$\mu^{-r} > \mu^{-s}$$

Это значит, что сообщение s имеет больше времени, которое может в очередях проводить:

$$\bar{T}^{r}_{s}(\mu^{-r}) > \bar{T}^{s}_{s}(\mu^{-s})$$

и

$$\bar{T}^{r}_{w} = \bar{T}^{r}_{q\,\text{крит}} - \bar{T}^{r}_{s}(\mu^{-r})$$

$$\bar{T}^{s}_{w} = \bar{T}^{s}_{q\,\text{крит}} - \bar{T}^{s}_{s}(\mu^{-s})$$

получим:

$$T_w^s > T_w^r \qquad / 37 /$$

Пусть сообщения r и s поступают в узел отправки /0-ой узел по пути/ и сразу могут обслуживаться. Вычисляя коэффициенты приоритета получим:

$$P_r^r < P_r^s \qquad / 38 /$$

Соответсвенно неравенству / 38 / сообщение r обслуживается впереди сообщения s, так как имеет меньше времени для ожиданий на каналы пути /это отражается в неравенстве / 37 //.

ж/ при передаче сообщения счетчик остаточного времени установится на новое значение:

$$T_{q\,ост}^j(t_{i+1}) = T_{q\,ост}^j(t_i) - \left((t - t_i^o) + t_{S_{i\to i+1}}^j\right) \qquad / 39 /$$

где $t_{S_{i\to i+1}}^j$ – время передачи j -го сообщения от i -го узла в $i+1$ -ый узел /время обслуживания каналом, соединяющим узлы/ пути.

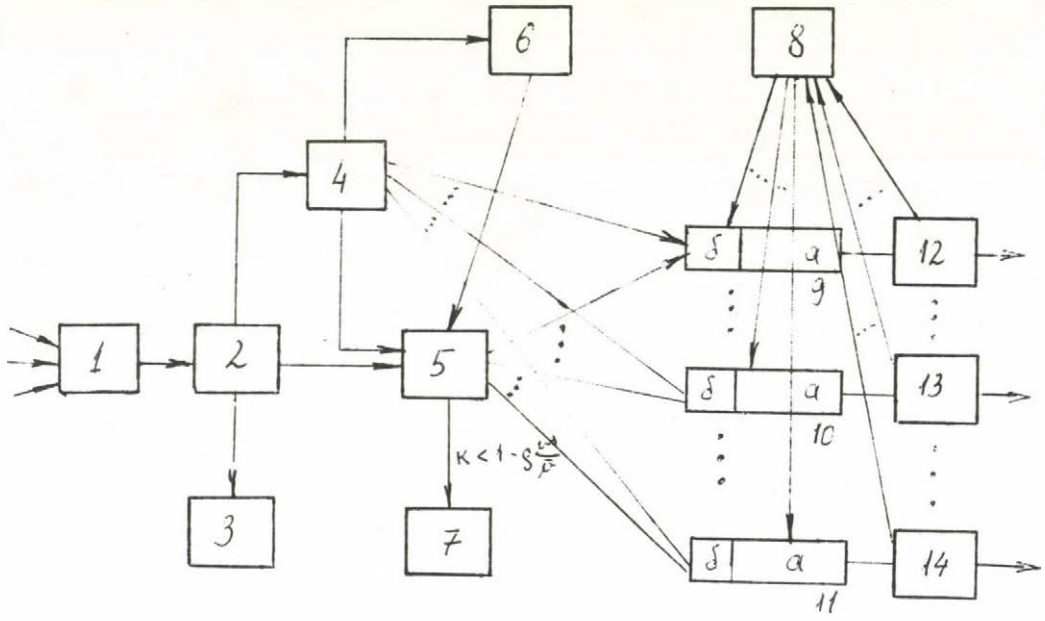
В i -ом узле значение $T_{q\,ост}^j(t_i)$ запоминается до получения положительного отклика об успешном приеме j -го сообщения в $i+1$ -ом узле. Сообщение вынимается из группы сообщений, составляющих очередь перед передачей в данное направление. В случае получения отрицательного отклика, j -ое сообщение снова ставится обратно в очередь со старым, запомненным значением $T_{q\,ост}^j(t_i)$. Это значит, что алгоритм выбора сообщения на передачу не сделает разницу между сообщениями, ждущими передачу и сообщениями, ждущими повторную передачу. Нет необходимостиразличать сообщения на основе этого, различие между сообщениями устанавливается по коэффициенту приоритета, показывающему абсолютное положение срочности сообщения. Коэффициент приоритета сообщения не учиты-

вает предысторию сообщения, только степень возможности своевременной доставки.

При передаче сообщение становится неактивным элементом очереди, т.е. условно стирается из очереди. Под действием обратно приходящего отклика сообщение

- действительно стирается из очереди - положительный отклик,
- активизируется в очереди - отрицательный отклик.

Нарисуем функциональную блок схему узла, работающего приоритетным методом обслуживания:



где I – прием сообщений,
2 – анализ сообщений,
3 – выход из сети сообщений с адресом данного узла,
4 – обработка и составление служебных сообщений,
5 – определение направления передачи и определение значения K ,
6 – таблица маршрутизации,

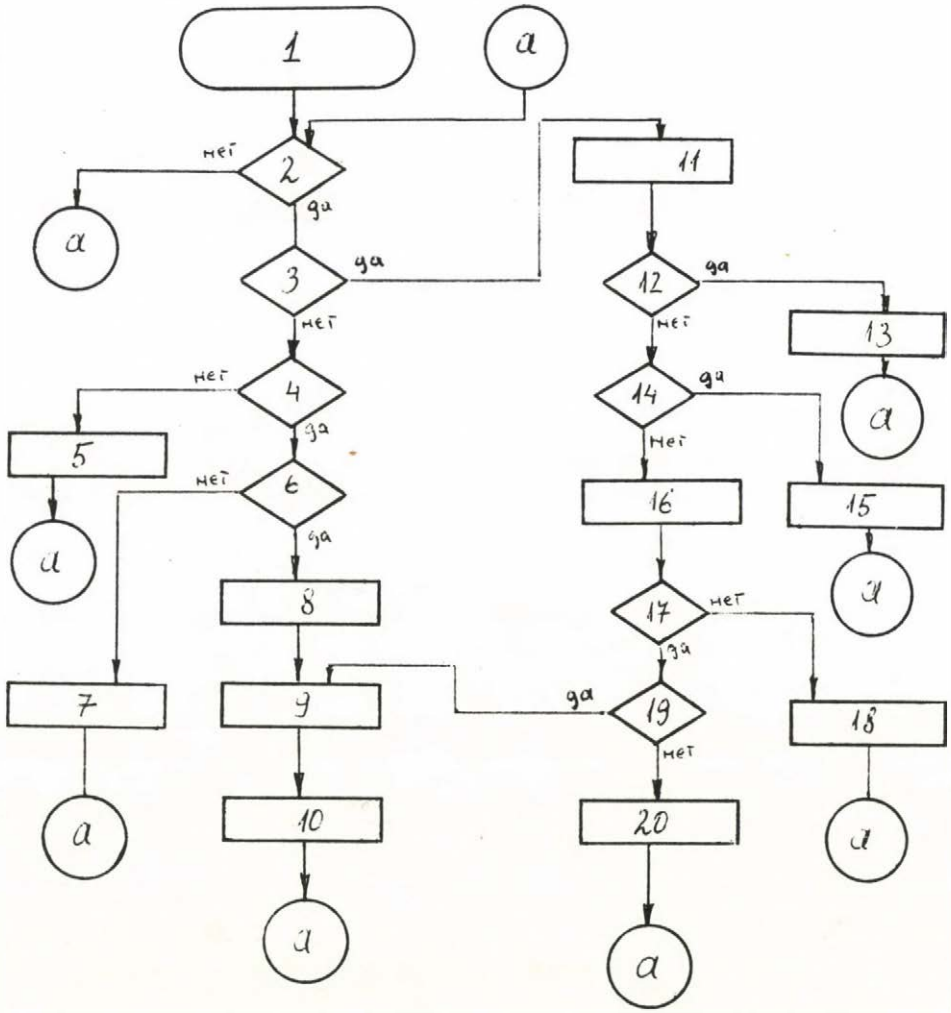
7 – специальное действие,

8 – управление очередями, выбор обслуживаемого элемента,

9, I0, II – очереди: а/ активные элементы, б/ неактивные элементы,

I2, I3, I4 – передача сообщений.

Нарисуем блок схему работы узла:



где 1 – старт,
2 – прерывание?
3 – прием?
4 – конец передачи в некотором направлении?
5 – обработка прерывания, не относящегося к передаче или к приему сообщений,
6 – есть активный элемент в очереди данного направления?
7 – сброс прерывания,
8 – выбор сообщения для передачи в данное направление /с минимальным значением P_r /,
9 – установка счетчика остаточного времени передаваемого сообщения,
10 – старт передачи сообщения,
11 – прием, обработка прерывания от приема,
12 – адрес адресата принятого сообщения равняется адресом данного узла?
13 – выход сообщения из сети передачи данных,
14 – служебное сообщение?
15 – обработка прибывших служебных сообщений,
16 – определение направления передачи принятого сообщения, вычисление $\overline{t_{обсл}}$ и K ,
17 – $K \geqslant 1+\delta_{\overline{P}}$?
18 – специальное действие,
19 – есть активный элемент в очереди данного направления?
20 – установка в очередь направления.

Литература

1. Kleinrock L.: Communication Nets; Stochastic Flow and Delay. "McGraw-Hill Book Company", 1964.
2. Kleinrock L.: Queueing Systems. v.1.,v.2. "John Wiley and Sons, Inc.", 1975,1976.
3. Пятибратов, А.П.: Вычислительные системы с дистанционным доступом. М. "Энергия", 1979.
4. Позин, И.Л. – Щербо, В.К.: Телеобработка данных в автоматизированных системах. М. "Статистика", 1976.

5. Советов, Б.Я.: Основы построения АСУ. Л. изд. ЛГУ, 1975.
6. Выставкин, Я.П.: Сети обмена информацией между ЭВМ. М. "Наука", 1975.

Address:
Computer Center for Universities,
H-1093 Budapest, Dimitrov tér 8.

The Use of Invariants in the Specification of Real Time Control Systems

Presented at the Visegrad Conference on the Theory of Operating Systems, Jan 1980.

S. J. Goldsack, Dept. of Computing and Control
Imperial College
London

Abstract

This paper investigates the use of invariants for specification of real-time control systems. Such systems can be veiwed as systems of communicating processes, each process responsible for maintaining some condition which is a design requirement at the application level. The plant itself obeys natural laws which can also be expressed in terms of the equations of motion, the eternally true conditions maintained by the laws of Physics. The use of an appropriate formulation in terms of invariant conditions appears therefore to offer an approach to specification which can embrace the specification of both the plant behaviour, and the requirements of the control engineers.

Introduction

It seems appropriate to start this paper with a quotation from a classic paper on General Systems Theory, by H. A. Simon.

> "By appropriate recoding, the redundancy that is present but unobvious in the structure of a complex system can often be made patent. The most common recoding of the descriptions of dynamic systems consists in replacing a description of a time path with a description of a differential law that generates that path. The simplicity resides in the constant relation between the state of the system at any time, and the state of the system a short time later."
>
> H. A. Simon, 'The Architecture of Complexity'
> Proc. Amer. Phil. Soc. 106, No6 1962

I hope to show that we can indeed derive specifications of dynamic and sequential control systems in terms of invariant assertions, which can constitute both semantic specifications of software processes, and specifications of the physical laws which control the behaviour of the external plant.

The paper is intended to be a contribution to the methodology of specifying computer systems for real-time control, which has formed the work of a Special Interest Group of The European Workshop on Industrial Computer Systems. At the recent Berlin Conference on Real time systems, Halling(1) presented a paper which emphasised the importance of the application level specification in the design and construction of such systems. Typically the design of a complete system takes place through a number of levels, each a specification of the system at some level of detail. (See fig. 1). The transition between the specification at one level and the next lower (more detailed) level constitutes a design step. Each level is more abstract than the one below it, in the sense that it concerns itself more with a specification of what is to be achieved, less with how it is to be done.

The whole design process starts from a specification of the application level;

at this point there is a dialogue between the plant engineers who determine the control requirements and the software engineers who understand the operation of computer systems. The two groups may use different languages to describe their systems- they have probably even been trained to think in different ways about their problems. There is great danger of misunderstanding at this stage, and we therefore consider it important to develop a well-understood formal approach to the specification of the requirements of the application level.

The specification should be

a) Abstract. i.e. it should specify what is to be achieved, not how it is to be achieved. Indeed, at this stage it should not even assume that a computer system is to be used. It is truly a specification of the application and its control requirements.

b) Formal. It should be expressed in a language which is unambiguous, and understandable by both parties to constrain their way of thinking about the system. The language should have a well defined semantic interpretation.

c) Hierarchic. In view of the well-known limitations of the human mind to comprehend complexity, the approach should preferably allow a system to be specified in terms of sub-systems, which are capable of being themselves specified in the same terms, but in greater detail.

In the paper referred to (1) three classes of control system were distinguished. First the class of problem which forms the domain of classical control theory, the dynamic control systems. Secondly there is the class of discrete variable, or sequential control systems. Lastly there is a class of system, albeit less well defined at present, which is 'data-oriented' being concerned with controlling the flow of data, maintaining a data-base and making control decisions based on complex data conditions. It was recognised that there are already available specification tools in each of these areas taken separately, but there appears to be no formal method of specifying systems involving a mixture of different types, the situation which most often occurs in modern automated plants.

The present paper aims to find some unity in the specifications by recognising the importance of system invariants. It so happens that the literature of computing already contains papers which relate to invariants in each of three types of system which appear to correspond with the above, so the use of such invariants may guide us towards a specification method meeting our needs.

Invariants in Software

The usefulness of invariants in the definition of the semantics of a program are well known from the work of Hoare (2), Dijkstra (3) and others. A 'While' program

```
while B do S
```

has an invariant condition of the data which can be expressed in an assertion $\{P\}$ such that, given that

$$\{P \wedge B\}\ S\ \{P\}$$

then

$$\{P\} \text{ while } B \text{ do } S \{P \wedge \neg B\};$$

Dijkstra has shown that such invariants can be a great aid in the invention of algorithms.

For example, a square root algorithm can be based on the invariance of the proposition P: $y = x^2 + r$

```
x:=0; r:=y; {y=x^2+r}
While |r| > eps do
"adjust x and r under the invariance of P"
```

Assuming that the adjustments do eventually achieve the condition for termination, then after termination we have

$$\{y=x^2+r \wedge r \simeq 0\} \quad \text{which implies } \{x \simeq \sqrt{y}.\}$$

It is clear that as well as the invariant the condition for termination plays an essential part in the semantics of the repetition. In a system process however, there is no concept of termination. We are concerned with what is achieved during the execution of each cycle of the program not the effect of completion of some finite number of repetitions. The expression of this effect lies in the invariant of the process body, the statement of what holds true whenever the program steps are completed. Thus we see that the invariant has grown in importance to be the main element in the semantic definition of the process.

System Invariants

A system consists of a collection of components each reacting to influences from other components and performing tasks allotted to it so as to satisfy some overall system laws. Naturally each element of the system must maintain its own part of the laws, so the concept of a system invariant is a natural part of the specification of a system component. The invariance reflects the constancy of the system laws with respect to displacements of the origin of time.

There are three main cases of system invariant discussed in the literature.

a) Hoare (4,5) discusses the integrity of a shared data structure in a system of concurrent processes. He shows that each user of such a structure must guarantee to conserve the truth of an invariant assertion, capturing the essence of the properties which characterise the object modelled by the data structure. The invariant is central to the proofs of correctness of monitors and abstract data types.

b) Kramer (6,7) showed that a useful concept in the construction of distributed systems is that of a stable module which continually monitors the state of the variables in its scope, and reacts when values stray outside some acceptable region of state space, in such a way as to bring about the necessary actions to restore the system to an acceptable state. The behaviour of the module is

therefore characterised by what Kramer calls a Quiescent invariant, the specification of that set of states which do not lead to corrective actions.

c) The present author (8) noted the important role played by an invariant in the semantics of cyclic operations, and was led to ask what is invariant in a control algorithm, simulating the effect of a controller in a classical control system. The answer proved rather illuminating. We shall see that it is closely related to the transfer function of the controller which it simulates.

In the following sections we shall examine these cases in more detail. It is in fact convenient to discuss these in the reverse order.

A Classical Feedback Control System

Consider the simple control system shown in fig. 2. The effect of the controller is simulated in a computer control system by an algorithm which repeatedly samples the state Y(t) of the plant variables and compares it with the 'set point' value D(t) to obtain the error value e(t). This is then used together with earlier values of e to compare U(t), the control to be applied to the plant. Since the control is maintained indefinately we can write the cycle of actions

```
while true do
  begin
    wait(t_n);

    input(e_n);

    compute (u_n);

    output(u_n)

  end
```

The invariant of this loop is the recurrence relation between u and e_n, e_{n-1}, e_{n-2} ..... established at each cycle by the step 'compute(u_n)'. In the notation used by control engineers this recurrence relation is expressed in operational form in terms of the transfer functions H of the controller. Following their practice we shall use an operational notation in which the operator E, operating on a plant variable X_n has the effect of advancing the time by one timestep, while I/E has the converse effect of retarding it

$$\text{Thus } EX_n = X_{n+1} \; ; \qquad \frac{1}{E} x_n = x_{n-1} \; ;$$

If the algorithm is linear the use of the operator E permits the recurrence relation to be expressed in a time independent way as an algebraic expression in E and e, emphasising its invariant nature.

As a simple example, suppose the calculation in the step

'calculate(u_n)' was given by the program statement

$$u := \alpha u + e \qquad \text{- - - - - - - - 1}$$

implying that $u_k = \alpha u_{k-1} + e_k$. ($\alpha$ is a constant < 1)

The effect of the algorithm can therefore be written

$$u = \frac{\alpha}{E} u + e \qquad \text{- - - - - - - - 2}$$

which can be solved to read

$$u = He;\ H = \frac{E}{E - \alpha} \qquad \text{- - - - - - - - 3}$$

In the language of the control engineer H is the transfer function of the controller, mapping the input time sequence e_n onto the output time sequence u_n. Equation 3 also expresses, however, the invariant assertion of the repeated program.

Suppose now that we wish to obtain the relationship between the variables u and e holding just before the assignment of the new value to u. In this case the left hand side is the next value of u so the assignment implies

$$EU = \alpha U + e$$
$$U = \frac{1}{E - \alpha} e \qquad \text{- - - - - - - - 4}$$

However, this result can also be reached by applying Hoare's axiom of assignment to equation 3. The relevant assertion before the assignment is

$$\alpha U + e = \frac{E}{E - \alpha} e$$

which reduces to the form of eqn. 4.

This simple algorithm corresponds to the solution by Euler's stepwise integration of the differential equation

$$\frac{du}{dt} + (1-\alpha)u = e.$$

It easy to extend the method to cover more general cases of higher order controllers, corresponding to differential equations of higher order. Though it is interesting to note that Hoare's assertions can be expressed in operational form, the important conclusion here is that the relation between the input sequence and the output sequence is expressed by the invariant relation

$$u = He$$

where H is the transfer function of the equivalent controller.

In general, the physical laws of the plant are also described by differential equations, and so the relationship between the force U impressed on the plant, and the state variable y is also expressed by a transfer function. Thus the plant maintains the invariant relation

$$y = Gu$$

which expresses the laws of physics.

Let us also note inpassing that we can now calculate the closed loop transfer function between the input set point D and the plant output y as

$$y = \frac{GH}{1 + GH} D \qquad - - - - - - 5$$

So the complete system maintains the invariant $y = H^1 D$ where $H^1 = \frac{GH}{1 + GH}$.

It appears that specification of dynamic control systems in terms of invariants has led us to a specification method which is a re-interpretation of the approach used by control engineers for decades. It is formal, abstract and complete. In view of eq^n 5 it is also hierarchic.

Discrete Variable Control Systems

We turn now to the second class of control systems, which is characterised by a set of discrete actions, and a set of conditions under which these actions must ensue. Typical of this type of situation is the start-up sequence in a plant, where it is prescribed for example that certain switches or valves must be actuated only when the plant has reached a certain state.

Sequential switching systems, such as relay systems, are also typical of this class of control, and there are some well established specification methods, such as ladder diagrams. There is also a German DIN standard relating to the specifying of such systems. When the permitted sequences involve complex synchronisation conditions, Petri-nets can also be used in specifying and validating such systems. However, none of these approaches seems to contain the necessary information about both the conditions and the nature of the ensuing actions to constitute a complete specification method.

Kramer's Stable Modules

Kramer introduced a notation for his modules which is closely related to Dijkstra's guarded commands. The following outline does not do justice to the full power of the notation, which makes provision for a hierarchy of enclosed modules. We consider here only the simplest type of module, which defines a set of actions, together with the set of conditions under which each action should take place.

A very simple example, given below, models the behaviour of an S.R. flip-flop in hardware

```
SR Switch ports (input S,R :bit; output Q: bit)
        begin Q=0        {creation condition}
          action
           set:          begin          S=1∧ R=0∧ Q=0
                           Q:=1
                         end      {Q=1}
           reset:        begin          R=1∧ S=0∧ Q=1
                           Q:=0
                         end      {Q=0}
          quiescence     (S=0∧ Q=0) ∨ (R=0∧ Q=1) ∨ (S=1∧R=1)
        end false        {termination condition}
```

The creation condition (Q=0 in this case) defines the initial state of the module. The assertions at the start of each action are the conditions under which the action should occur. Since the relevant condition is true at the start of the corresponding action, it is also the precondition for the ensuing program segment. These assertions correspond to the guards in Dijkstra's guarded commands. If several guards are simultaneously true, then an eligible action is selected non-deterministically for execution. Activity will therefore continue until all guards are false. The module communicates through port variables which are typed. In this case they all have type bit defined by

$$\underline{\text{type}}\ \text{bit} = (0,1)$$

The module is 'quiescent' when there are no eligible actions, ie. no true guards. The behaviour is similar to the do construct of Dijkstra, except that quiscence does not imply termination. The activities will start up again if a change in the value of the input variables satisfies one or more of the guards again.

If guards for actions $A_1, A_2, \ldots\ldots\ldots A_n$ are $P_1, P_2, \ldots\ldots P_n$ then the condition of quiescence is $\{ P_1 \wedge P_2 \wedge P_3 \ldots\ldots \}$ which reduces in this case to the condition shown in the module text. The third term $(S=1 \wedge R=1)$ should be regarded as an input error condition. Either this situation is fully excluded by the module producing the inputs, or it could be used as an additional guard for an error action, or alternatively it could be used as the termination (or abortion) condition for the module.

A somewhat similar concept was introduced by Halling and Haase, (9) but the semantic interpretation was somewhat different. In their case concurrent actions occur if several guards are simultaneously true. Here we shall continue to use Kramer's version.

The question now arises of how such a module communicates with others in the system. This question is discussed at length in Kramer's thesis, but the most satisfactory view is that communication occurs only in the quiescent state.

Perhaps a useful analogy is with a hardware 'black box' in which communication is controlled by a 'busy/ready'flag. The module will receive inputs, or produce meaningful outputs only when the flag signals non-busy. The internal actions of the module are therefore entirely invisible from outside.

We shall return to the question of communication between modules and processes a little later.

Relationship Between the Continuous and Discrete Models

It is of interest to note at this stage the difference between the two cases discussed above. In the first the action is to input variables from the environment synchronously, at times selected by a clock, and output the computed control signal. In the other case the input of data is asynchronous. That there is in fact little difference beyond

this can be demonstrated by making the interaction with the clock explicit, and writing

```
controller port input e: status; t:time output u:control
        begin   n=0 ∧ τ= timestep
          action
            compute:  begin  t ≥ n * τ
                        n:= n+1
                        compute (U_n);
                      end  {u=He}
          quiescence     (u=He) ∧ (t < n*τ)  {but see footnote}
          end false
```

which has the same effect as the first model.

To give further emphasis to the essential unity of the concepts involved, suppose we replace the controller of fig 2 by that required for a 'bang-bang' servo, in which the actuator is driven to a constant level -a if the error signal exceeds some limit ε, and to a level +a if the error is negative and greater than that limit. If the error is within the dead zone abs(e) < ε then the control signal is set to zero. This gives a very crude homing action on the set point e=0.

```
bang-bang control port input e:status output u:control

        begin
          action
            set_to_drive_up:        begin       e≤-ε ∧ u≠+a
                                      u:=a:
                                    end   {u=a}
            set_to_drive_down:      begin       e≥ε ∧ u≠-a
                                      u:=-a
                                    end   {u=-a}
            set_to_rest:            begin       abs(e) < ε ∧ u≠0
                                      u:=0
                                    end   {u=0}
          quiescence  (abs(e) <ε ∧ u=0)  ∨  (e≤-ε∧u=a)
                                         ∨  (e≥ε∧u=-a)
          end
```

The invariant is the truth of the desired functional relationship between u and e, which could also be expressed u= if abs(e)<ε then 0 else if e ≥ ε then -a else a.

Footnote:
strictly quiescence is (t < n*τ). The actual invariant is the stronger statement given here, since the actions cannot create states with (u≠He).

Interacting processes

I pass now to a discussion of the mechanisms of interaction between processes. For this I would like to refer to the work of Fitzwater and Zave (10), who have developed a formal theory of processes in terms which seem to fit well with our present needs.

In the model introduced by these authors, a process is considered to be a pair $(F.\sum)$. $\sum$ is the state space in which the process state is defined, while F is a function which takes a state vector as argument, and returns a new state vector. Successive applications of F generates the development of the process in time. The evaluation of F may involve the evaluation of what the authors call an exchange function. Such a function takes arguments which are drawn from the set constituting the state variables of the process. It can have quite explicit side effects: namely it may plant new values in the scope of some communicating process, at the same time returning values which have been supplied by the communicating process. If the communicating process is not ready to make the exchange of values, then the evaluation of the exchange function, and hence of the function F may be delayed. Of course the data transferred to the communicating process are also received by it as values returned by its evaluation of a similar exchange function. The interaction of processes in this way constitutes a rendezvous, in the sense used in the Ada language,(11) a communication of information between willing partners, who are each ready to exchange information. Let us now assume that readyness to communicate exists at quiescence. Then a pair of communicating processes will guarantee to exchange values only in sets which satisfy the invariant condition. Of course it is probable that the exchange of values will destroy the invariant of one or other or both partners.

With this view of processes, process communication and invariants, let us now look at fig 3a. Here we have three processes corresponding to an operator, or control room, a controller, defined by a transfer function H and a plant object whose physical properties are defined by G. Between the controller and the plant are exchanged the values of Y and U respectively, each conforming to the invariant condition defined at quiescence. Between the control room and the controller, the exchange is of the values of the current set point, as required by the operator and the status of the control Y. The invariants which are relevant here are on the one hand the informal statement "D is the currently defined set point" and on return the invariant relation Y=H'D derived earlier. We can, of course, draw a box around the combined pair of processes, controller and plant, and treat them as a single process, exchanging values with the control room and preserving the latter invariants only. Thus, in the domain of classical control systems the approach meets the requirement of hierarchic decomposition. We could of course replace the **controller** by the Bang-bang controller without spoiling the picture, though there would not now be so simple a result for the closed loop transfer function.

A Resource Sharing Example

Kramer describes a solution in terms of stable modules of a resource sharing system in which two users of a resource interact with a resource allocator, which arbitrates btween them in case of contention. It therefore constitutes a resource monitor in which user requests and allocation are represented at a more primitive level than is customary in recent programming languages.

A user has a state rr, of type boolean, which means "resource requested". It receives its allocation through a variable ra, "resource allocated", also boolean. The user has two activities "processing without resource" and "processing with resource", and its quiescent state is $(rr \wedge \sim ra) \vee (ra \vee \sim rr)$, the two terms representing the situations in which it is either waiting to receive the resource, or waiting to return it.

The allocator, for its part, tries to arrange that the resource is available to anyone requesting it, and will immediately allocate it if it is free. Using suffices to distinguish the users, it maintains the invariant

$$(rr_1 \wedge ra_1 \wedge \sim ra_2) \vee (rr_2 \wedge ra_2 \wedge \sim ra_1) \vee (\sim rr_1 \wedge \sim ra_1 \wedge \sim rr_2 \wedge \sim ra_2)$$

the union of three situations in which it should not make any change to the current allocation.

A user need not be aware of the other user competing for a share of the resource; so far as he is concerned the allocator is maintaining the partial invariant

$$(rr_1 \wedge ra_1) \vee (\sim rr_1 \wedge \sim ra_1)$$

Fig 4 shows the set of processes user 1, user 2, and allocator, with their relevant data exchange paths. In general terms it is clear that these invariants give a helpful view of the way in which these modules appear to their neighbours, which amounts to a partial specification of their behaviour as system components However, there are two relevant comments.

a) The exact mechanism for communication needs further investigation. It is easy to see that the synchronised exchange of values suggested in the previous case is not sufficient, and a more carefully formulated definition of the communication mechanism is required.

b) The specification provided by the invariants is not complete. The user module, on receiving its allocation, enters an active phase "processing with resource". The nature of this processing is not specified. It is, however, probable that the processing involves a shared data resource. In that case we are indeed considering a monitor, and the data invariant used by the specification of monitors will be relevant.

It should be possible to unify this specification with those used in current work on abstract data types.

These questions are receiving active study.

Conclusions

A control system can be regarded as a collection of processes which communicate in a disciplined way. The state of a process is visible to others only through communication, which can be arranged to occur at a time when a invariant condition is satisfied. The invariant therefore specifies the semantics of the process, as it is seen by its neighbours.

This approach includes the specification both of the controller processes and of the plant objects themselves.

References

(1) H. Halling, V. Haase, S. J. Goldsack, J. Kramer
"A Step Towards Application Oriented Specifications" Berlin Conference "Real Time Data 79" October, 1979. (Also Report No. 39, Research Centre, Technische Universitat Graz).

(2) C. A. R. Hoare
"An Axiomatic Basis for Computer Programming" CACM 12, 10, October 1969.

(3) E. J. Dijkstra
"A Discipline of Programming" Prentice Hall 1976.

(4) C. A. R. Hoare
"Proofs of Correctness of Data Representations" Acta Informatica 1 1972.

(5) C. A. R. Hoare
"Monitors : An Operating Systems Structuring Concept" CACM 17, 10, October, 1974.

(6) J. Kramer and R. J. Cunningham
"Towards a Notation for the Functional Design of Distributed Processing Systems" Proceedings of 1978 Industrial Conference on Paralled Processing.

(7) J. Kramer,
PhD thesis, Imperial College, London. 1979.

(8) S. J. Goldsack
"Semantic Definition of Linear Control Algorithms" Computer Journal 21, 381, 1978.

(9) H. Halling and V. Haase
"Description of Real Time Applications using the Guarded Commands Concept" Report 75 Universitat Karlsruhre 1978.

(10) D. R. Fitzwater and P. Zave
"The Use of Formal Asynchronous Process Specifications in a System Development Process" Texas Conference on Computing Systems November 1977.

(11) Ada Preliminary Reference Manual, published in "SIGPLAN NOTICES" Vol. 14 June 1979.

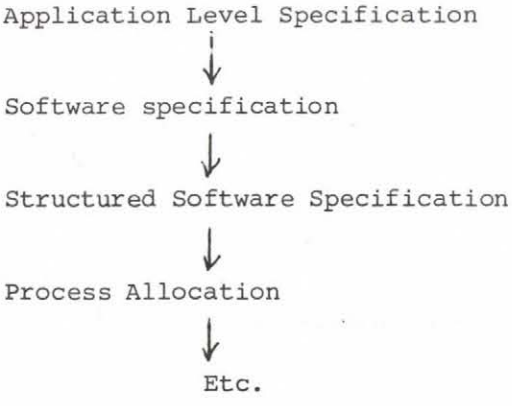


Fig 1 Typical Sequence of Design Steps in Constructing a Real Time System

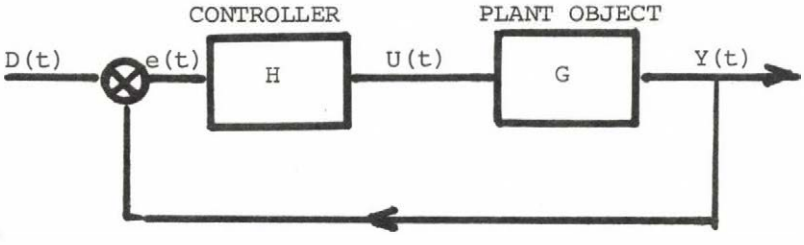


Fig 2 Simple Classical Feed-back Control System

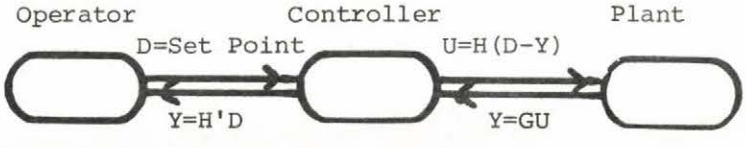


Fig 3a Control System as a Set of Processes

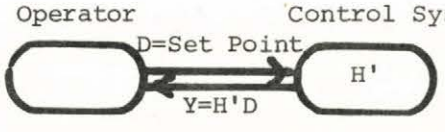


Fig 3b Closed Loop System viewed as one Process

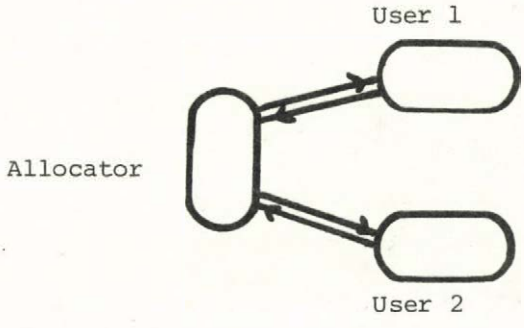


Fig 4

MODELS OF DISCRETE CONTROL SYSTEMS

by

V.H. Haase
(Austria)

Abstract: A methodology for the specification of discrete control systems is discussed. It is based on two types of models (abstract process control systems). We distinguish between action oriented models and data oriented models.

Action oriented models describe the abstract "task"-structure; the elements are conditions (i.e. predicates on the state of the systems) and actions (i.e. sequential processes).
Data oriented models contain the attributes of all process objects in a relational data base. Actions are formulated as transformation rights.
"Communication data blocks", the validity of which is precisely defined in space and time are used to combine various submodels.

1. Specification of process control systems

Digital computers allow the replacement of analog by digital control systems. Especially the use of microprocessors in distributed systems gives radically new possibilities in the design of process control systems. Hardware and software development are no more separate activities when distributed systems consisting of different types of processors are used. In projects of this kind different people: e.g. metallurgical engineers, electronics engineers and computer specialists are cooperating: they need a common language.

This paper discusses the use of models to specifiy requirements, and to design control systems for (industrial) real-time processes. These models shall be a common base for mutual understanding, and shall be applicable for various types of industrial processes, and for any architecture of the computer system to be used.

We distinguish between requirements specification, and the design process. Models for requirements specification describe the application: they form an implementation independent abstract control system. They are the starting point for the design phase where a model of the automation system: still abstract, but implementation dependant, is built. In practise many errors occuring in process control systems are caused by misunderstandings between the application specialists (who specify the requirements of the industrial process), and the automation engineers. We need a language which can be used for a functional description of all components of both the industrial process and the automation system. The language must be application

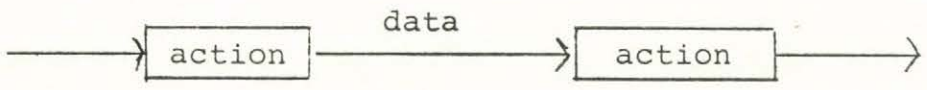
oriented (to be accepted by the industrial engineers), but also sufficiently formal so that it can be mapped into design decisions of hardware and software.

A similar technique can be found in block diagram languages for continuous (analog) processes. In the digital case each block represents a sequential (finite state) machine performing a well-defined function. A further important concept is "global control": communication and synchronisation of functions is performed by the the interconnecting network (not by the local blocks!). Functions and cooperation are specified separately. This level of requirements specification does not contain any informations about the implementation. These are taken into account during the next (=design) phase, possibly they may (iteration!) be used to adapt requirements later (e.g. cost, availability of modules etc.).

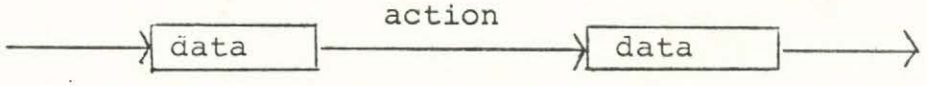
2. Types of processes and models

A model consists of objects and of relations between objects. It serves to facilitate to understand how the physical process works, and to describe it in a general formal way. It describes functions not the implementation.

Dependent on the type of problem we distinguish between action oriented models (for sequential processes):



and data oriented models (for discrete manufacturing processes):



2.1 Sequential industrial processes

Discrete events (changes in the state of the system) trigger actions which are executed in strict sequential order. Examples are startup- or shutdown-sequences, and checkout procedures. Mathematical analysis can use transaction nets (e.g. Petri nets) or predicate transformers. We shall discuss one of these approaches (the "PARCS") in more detail.

While automation systems for sequential processes can be programmed in various ways (e.g. ADA or PEARL), on the level of requirements specifications hardly any proven method exists. Graphic representations: relay ladder diagramms or "Funktionspläne" (DIN 40719/5) are used, but not in connection with formal analysis. Models should be verifiable in the sense that the fulfillment of requirements can be checked in a formal manner.

2.2 Discrete manufacturing processes

The objects within the industrial process are described using data structures. Typical examples are transportation, assembly-line and automatic chemical analysis. Apart from some applications of simulation techniques, and the use of queuing theory hardly any mathematical model exists. We shall try to work with an approach similar to Jackson's design method, but based upon relational data base models.

Data relations are (at any time) an image of the state of the system. They are specified by the application

engineer (e.g. in the form of invariants). Common resources of the industrial process are specified in the same way as data blocks, which can only be handled by one transaction at a time.

3. PARCs, an action oriented model

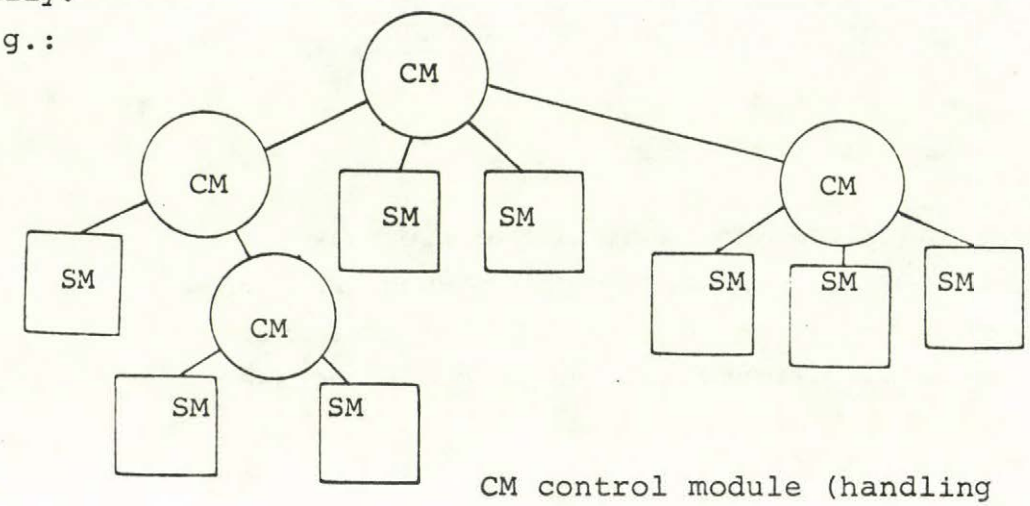
PARCs are a specification means from where task-oriented programms (e.g. written in PEARL or PROCESS-FORTRAN) can directly be derived. In the first step the problem is partitioned into sequential processes (actions), an action is defined as the set of statements between two points of synchronization. Synchronization is defined in the second step: it is determined by changes of the global state of the system.
In this model the control system (resp. parts of the control system) is specified as a set of pairs:

$$\left\{ \begin{array}{l} \text{"condition"} \longrightarrow \text{"action"} \\ \text{(or "guard")} \end{array} \right\}$$

Each pair defines a concurrent process. "Conditions" are predicates over the values of a subset of global variables, at any specific moment they are either true or false. "Actions" are started when the corresponding condition becomes true, they are executed sequentially without interruption (=virtual processors). Thus the specification of application oriented activities in the system (e.g. control algorithms) is strictly separated from the communication system (which links different concurrent processes).

The structure of the control system can be shown graphically.

E.g.:



CM control module (handling communication)

SM sequential module (actions= sequ.algorithms)

The sequential modules are autonomous parallel processes which communicate with other active modules only at starting and ending time via ("call-by-value") parameters.

The PARC-model has the following properties:

a. It corresponds to the "naive" understanding of plant engineers: "if" a state that is relevant for the application process has changed, "then" do something. The various if-then pairs are independent of each other unless explicitely specified in some condition.

e.g. temperature > 500 degrees → close-gas-valve
time = 18:00:00 → switch-on-lights

b. Semantics can be specified in the form of predicates over the state of the system before and after the execution of actions. This semantic analysis can even be extended to determine the behaviour of the process in real-time (= calculation of reaction times).

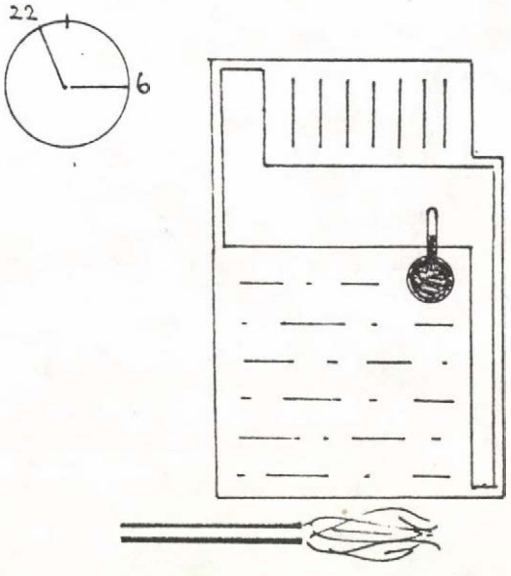
e.g. pardo {producer-consumer, 2 buffers}
p ≠ 2 → p:= p+1 {producer; exec-time = t_p}
□ c ≠ p → c:= c+1 {consumer; exec-time = t_c}
parod

wp(producer-consumer,t≤T) = t≤T-t_p-t_c-max(t_p,t_c)
(wp determines the latest point in time when the process: exchange of 2 records via a buffer has to be started if it must be finished prior to t=T; t=real-time, T=deadline).

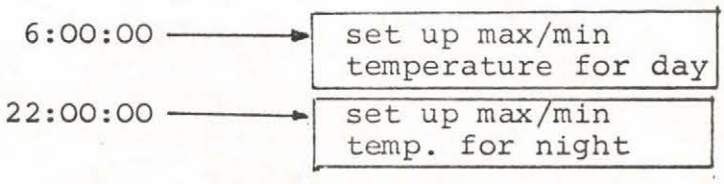
c. No prescription is given how sequential modules must be implemented; "hard-wired" elements may be used as well as assembly-language programs in a micro, or a RSX-11-FORTRAN task.

d. A drawback of this method is its inability to descibe resource sharing in an easy manner.

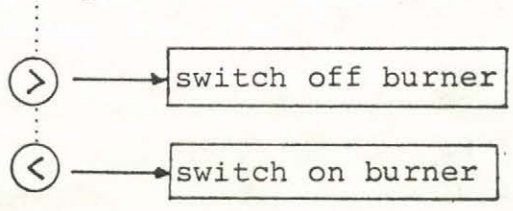
Example: Central heating system



control flow model:



measured temperature:



PARC-specification:

```
pardo heating
   time = 6:00:00 -> day-setup
□ time =22:00:00 -> night-setup
□ temp > max -> switchoff
□ temp < min -> switchon
parod heating
```

(PARC-specification continued:)

```
procedure switchon        ⎫
.                         ⎪
.                         ⎪   different methods
end                       ⎪   of hardware and/or
procedure switchoff       ⎪   software implementation
.                         ⎪   are possible
procedure day-setup       ⎪
.                         ⎪
procedure night-setup     ⎭
```

4. A data oriented model

In the case that a technical process is determined by sets of objects which are worked at rather than by action sequences (e.g.in manufacturing), models that are based on global data structures are to be preferred. (In the action-oriented approach complicated actions work with simple data, in this model simple actions work with complicated data.)

The user specifies data relations:

$$R \subseteq W_1 \times W_2 \times \ldots \times W_n$$

over the range of values of the variables ("attributes") W.

These relations are regarded dynamically, i.e. each attribute also has a range of validity in space and time. "Actions" are specified indirectly as "transformation rights" to change attribute values. They are parameters of attributes and are "triggered" by the change of values of (the same or other) attributes.

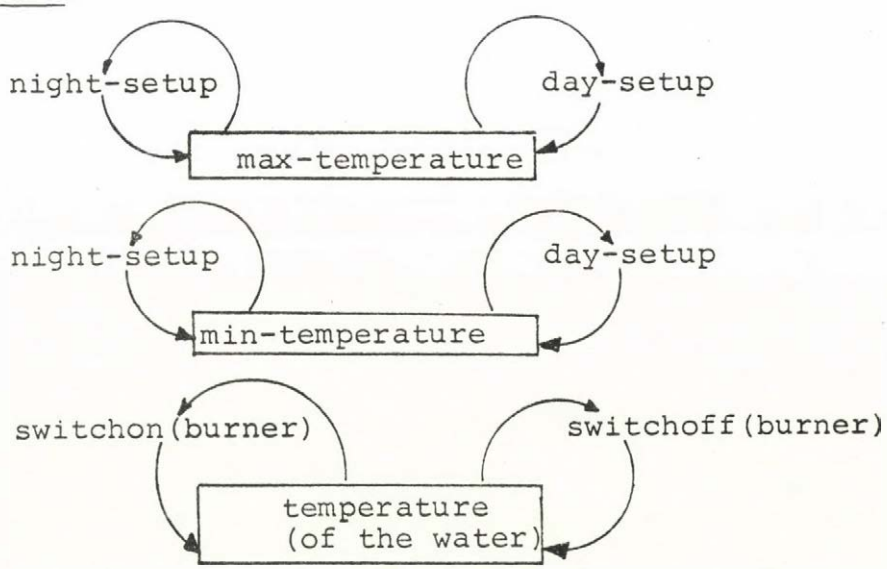
e.g.: relation: central heating
attribute: maximum temperature
value = 35 degrees
validity = time≥6:00:00
transformation = (night-setup)

.
.

This data oriented approach has the following properties:
a. It is related to a naive point of view too: an "object" with the following attributes may be changed as follows:
e.g. "maximum-temperature" by "night-setup" ;
"burner-state" by "switchon-burner" etc.
b. Correct synchronization and data-integrity can be guaranteed implicitely, if all data-transformations (= actions) are defined to be mutually exclusive. In the context of real-time this point nevertheless has not been investigated up to now.
c. It is left to the decision of the implementer how the data relations are represented in the machine: decision-table techniques are possible, as well as methods related to Jackson design.
d. The method is not suited to describe functional dependencies of actions, i.e. action sequences. (Thus the two models are in some sense complementary.)

Example: Central heating system

data flow model:



Data-oriented specification (tentative):

item::= name value validity transformation-rights(mode)

e.g.:

max	35.0	time 6:00:00 time 22:00:00	day-setup(update) night-setup(update)
temp	undef.	'true' (=always)	switchon(environment) switchoff(envir.)
...			

5. Open questions

These models have been used as specification tools for several small problems at University of Karlsruhe (simulator for distributed systems) and KFA Jülich (industrial robots). Large systems have not been tackled yet, so two sets of questions are not answered:
a. which criteria should be used to acchieve the best segmentation of the problem: are the "pairs":
global conditions — sequential actions resp.
data-structures — transformation rights
sufficient ?
b. how should communication between different subsystems (which may have been specified in different ways) be described? A feasible method could be based upon "communication-data-blocks" which are triples:
name value validity
where validity is a predicate on the state of the system (dependent on space and time).
Apart from the use in large problems also techniques for the verification of system-designs based on models like these have to be developped.

Nevertheless this approach seems to be promising to bridge the gap between formal and precise specification methods (which exist for mathematical problems like producer/con-

sumer-exercises, but are too complicated to be used in real world problems) and application oriented techniques (e.g. SADT, PSL/PSA, EPOS etc.) which are understandable but not in same sense precise.

6. Literature

W.-M.Dehnert/V.Haase: High Level Language Structures for distributed real-time programming. SOCOCO 79, Prag (1979)

E.W.Dijkstra: A Discipline of Programming. Englewood Cliffs (1976)

P.Elzer: Strukturierte Beschreibung von Prozeßsystemen. Dissertation Univ.Erlangen-Nürnberg (1978)

P.D.Griem: Approaching an easy-to-learn method of programming real-time parallel processes. IFAC/IFIP Workshop on Real-Time Programming, Paris (1976)

V.H.Haase: Spezification and Construction of Real-Time Programs with PARCs. Angewandte Informatik, Heft 5, 1980

V.H.Haase/H.Halling: Descriptions of Real-Time Applications using the Guarded Commands Concept. AFCET-Workshop on global description methods for synchronization, Paris (1977)

H.Halling e.al.: A Step towards Application Oriented Specifications. Real Time Data 79, Berlin (1979)

G.Hommel(ed.): Verfahren und Hilfsmittel für Spezifikationen und Entwurf von Prozeßautomatisierungsverfahren. KFK-PDV 154, Karlsruhe (1978)

M.A.Jackson: Principles of Program Design, London (1975)

R.Lauber: Prozeßautomatisierung I, Berlin/Heidelberg (1976)

D.T.Ross: Structured Analysis. IEEE TSE 3, 16-34 (1977)

G.Schlageter/W.Stucky: Datenbanksysteme. Stuttgart (1977)

author's affiliation:

Volkmar H.Haase
Institut für Informationsverarbeitung
Technische Universität
Steyrergasse 17
A - 8010 Graz / Austria

author's current address:

c/o Institut für Angewandte Informatik
und Formale Beschreibungsverfahren
Universität Karlsruhe
Postfach 63 80
D - 7500 Karlsruhe 1 / Fed.Republic of Germany

A FLEXIBLE MEASUREMENT SYSTEM FOR THE PICTURE PROCESSING BASED ON A PDP11/4o COMPUTER.

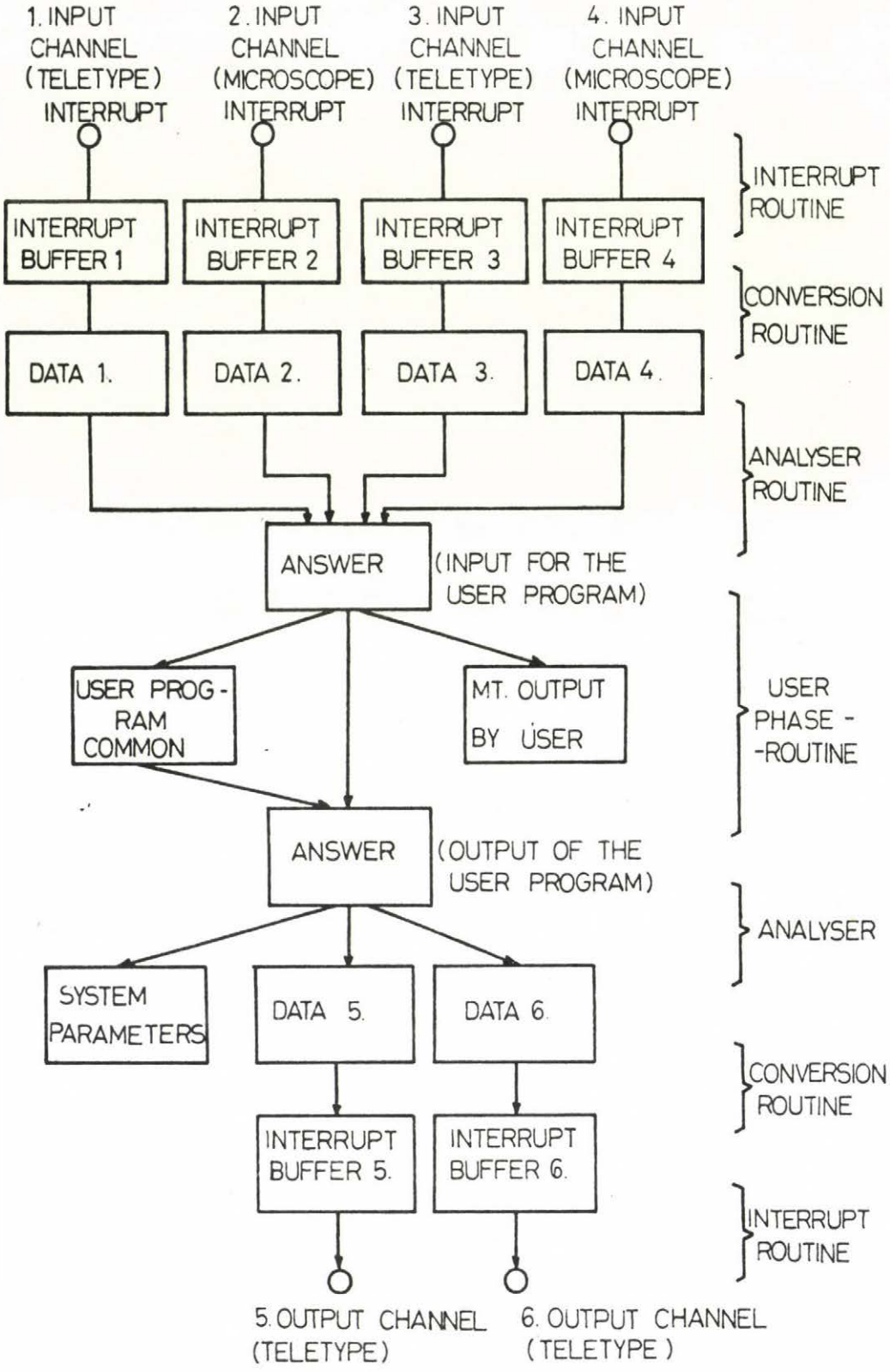
Agnes Holba, Hungary

INTRODUCTION

In our laboratory a set of evaluations of high energy measurement data has been going for years in different kind of international cooperations. For this reason a hardware system has been developed to increase the speed of measurement and data processing. Four apparates for measuring of coordinates are interfaced to a PDP 11/4o computer by CAMAC MOTOR CONTROL INTERFACEs 2.13, and a teletype or a display interfaced by CAMAC SERIAL INPUT/OUTPUT INTERFACE 3.o4-2 is given for each measuring equipment. The measuring equipments differ from each other in the form of data and in the method of the measuring, so the processing programs have to be also different for each measurement. Our software system has been developed for the on-line control of the measuring and the data processing.

THE PHILOSOPHY OF THE SYSTEM

The interrupts and the data are coming random from the different devices. During the first part of the processing the data are converted into the data form of a high level program language. For example the 24-bit integer data of CAMAC are converted into 32-bit floating point numbers. When a data set converted by the system is completed with an END-SIGNAL, the control is given for the suitable routine of the processing program to test and store them, to send message for the operator, and to give the control parameters for the system. The data transfer between the different arrays is shown on the picture 1.


1. INPUT CHANNEL (TELETYPE) INTERRUPT
2. INPUT CHANNEL (MICROSCOPE) INTERRUPT
3. INPUT CHANNEL (TELETYPE) INTERRUPT
4. INPUT CHANNEL (MICROSCOPE) INTERRUPT
INTERRUPT ROUTINE
INTERRUPT BUFFER 1
INTERRUPT BUFFER 2
INTERRUPT BUFFER 3
INTERRUPT BUFFER 4
CONVERSION ROUTINE
DATA 1.
DATA 2.
DATA 3.
DATA 4.
ANALYSER ROUTINE
ANSWER
(INPUT FOR THE USER PROGRAM)
USER PROG - RAM COMMON
MT. OUTPUT BY USER
USER PHASE - -ROUTINE
ANSWER
(OUTPUT OF THE USER PROGRAM)
ANALYSER
SYSTEM PARAMETERS
DATA 5.
DATA 6.
CONVERSION ROUTINE
INTERRUPT BUFFER 5.
INTERRUPT BUFFER 6.
INTERRUPT ROUTINE
5. OUTPUT CHANNEL (TELETYPE)
6. OUTPUT CHANNEL (TELETYPE)

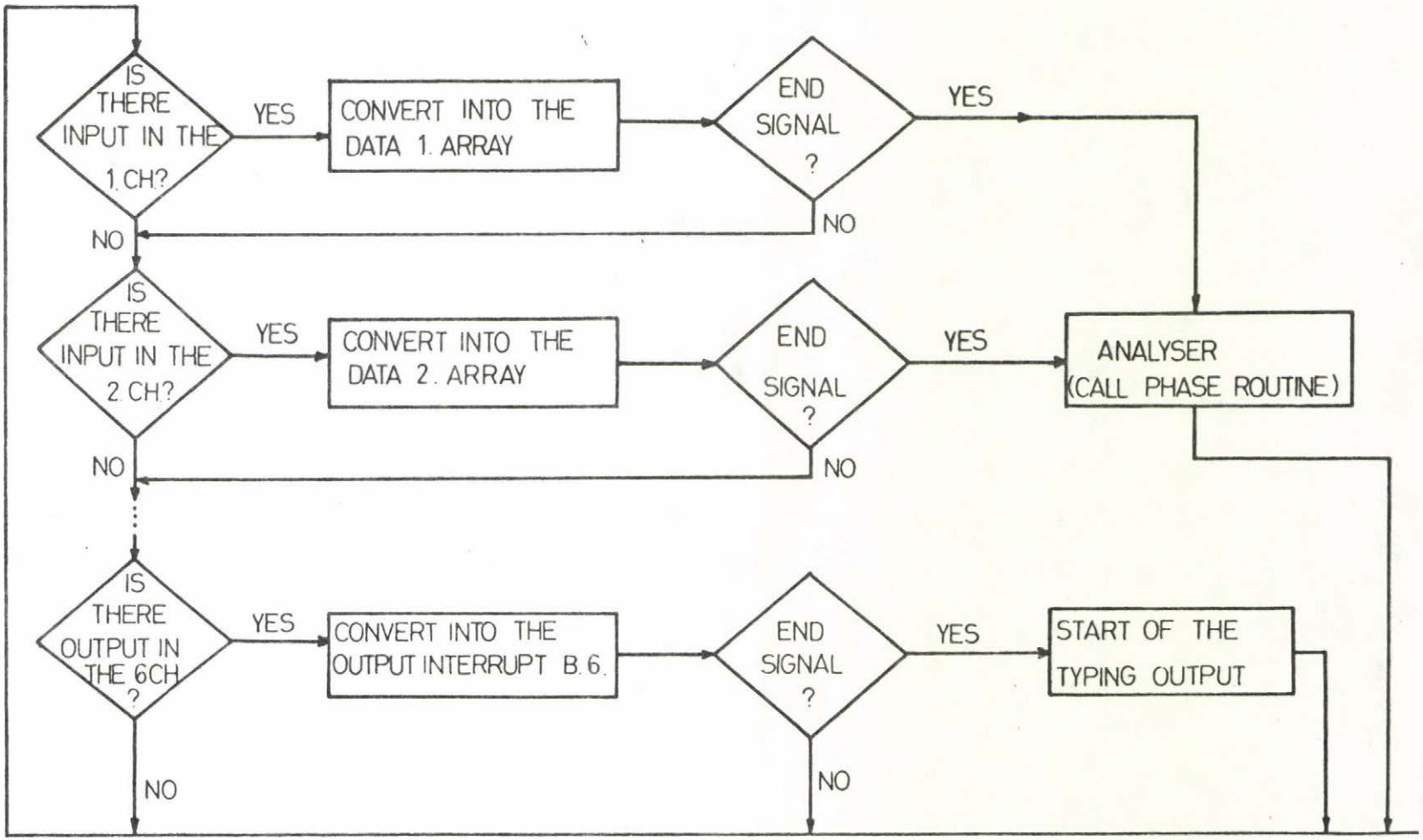
Picture 1.

THE STRUCTURE OF THE SYSTEM

The system program consist of three parts. In the first part there are all interrupt service routines, they fill the interrupt buffers. These routines are always active unless the buffers are full. From the interrupt buffers the data are converted into the DATA arrays by the converter routine. The converting of an END-SIGNAL starts the analyser routine. This routine tests the data syntactically, puts them into the ANSWER common and calls the suitable user routine. The user routine can overwrite the ANSWER. After return from the user routine the analyser tests the output caracter string being in the ANSWER, puts it into the DATA array of the suitable output channel, gets the control parameters coming from the user subroutine, and gives the control to the converter routine,wich converts the data from the DATA array into the interrupt buffer. The scheme of the converter connecting with the analyser is shown on the picture 2.

THE PROCESSING PROGRAMS

The processing programs are written by the user. They can be written in any high level program language, but a few rules have to be taken into consideration. These are the following. The user programs have to consist of subroutines, so-called phase routines, wich can be called one after another. A phase routine has to correspond to a step ofthe measurement. The data coming from the different devices are given for the phase routine through the ANSWER. Depending on the succes of the data test the phase routine has to decide on the following measurement step, to give the step number for the system and to send the message character string for the operator. These data get into the system through also the ANSWER. The number of the following measurement step can be pointed any of the


IS THERE INPUT IN THE 1.CH?
YES
CONVERT INTO THE DATA 1. ARRAY
END SIGNAL ?
YES
NO
NO
IS THERE INPUT IN THE 2.CH?
YES
CONVERT INTO THE DATA 2. ARRAY
END SIGNAL ?
YES
ANALYSER (CALL PHASE ROUTINE)
NO
NO
IS THERE OUTPUT IN THE 6CH ?
YES
CONVERT INTO THE OUTPUT INTERRUPT B.6.
END SIGNAL ?
YES
START OF THE TYPING OUTPUT
NO
NO

Picture 2.

phase routines,wich have been linked to the system. The phase routines can call any other subroutine,and they can directly use the allocated pheripheries. All statements permitted by the compiler can be used in the routines. The different measurements can use the different set of phase routines, but any routine can be common as well. A phase routine can lightly be changed by linking the new routine instead of the old one. So the system can be fitted well for the variable requirements.

REFERENCES

1. CAMAC
   EURATOM report EUR 4looe,1972
2. Gémessy Tibor: Az MS-IV ellenőrző/scanning/-mérő vetitő bemérése.
   KFKI-1979-62 Budapest, 1979
3. V.Zacharov: Data transfer in on-line systems.
   Proceedings of the 1978 CERN school of computing.
   Geneva 1978.

CENTRAL RESEARCH INSTITUTE FOR THE PHYSICS
OF THE HUNGARIAN ACADEMY OF SCIENCES
H-1525.Budapest POB 49.

TRANSFORMING RECURSIVE STRUCTOGRAMS INTO DO-WHILE STRUCTOGRAMS

László Hunyadvári
L. Eötvös University, Dept. of Comp. Sci.

1) Introduction

The aim of this work is to give a recursion removal method for a special type of recursive procedures described by recursive structograms.

Recursion elimination may be necessary in such cases, when we can dessign an algorithm naturally by using recursion, but in our programming lauguage there is no possibility for explicit recursion. This is the situation at that problems, where the data structure is defined by recursion, for instance at tree-traversal algoritmus. The recursive program is well structured (the data and program structures are in correspondence) and its correctness is clear or can be proved easy. If we want to solve this problem directly by a nonrecursive program, then its structure is generally more complicated and can be proved with difficulty.

Another reason for the research of reversion elimination techniques is the following: if we remove the recursion by hand, then the transformed program is usually faster and needs less space than the implementations provided by general mechanism of existing compilers.

The recursion removal problem is interesting from theoretical point of view too. In [1] and [2] Strong and Walker characterize those recursion procedures which can be transformed into equivalent iterative forms without using stacks. In [3] Partsch and Pepper and in [4] Bird deal with special kinds of recursive procedures which can be rewritten into iterative form by the help of a stack.

In this work we study a more general type of recursive procedures, which are defined by recursive structograms. They are transformed into a special flowchart form, the Chapin-chart form [5] (or by an other name DO-WHILE structograms).

The principle of recursion elimination is well - known - the chain of recursive calls is simulated by a loop wich haudles a stack memory. At a new call we push the information, necessary for continuation (namely, the values of local variables of calling level and other auxilary information), into the stack. After storing these return-informations we can execute our recursive procedure from the beginning, with the suitable actual parameters. If the running level is finished we take out the return-information from the stack, pass the results from the formal parameters to the actual parameters and continue the calling level. If the stack is empty, the current level is the topmost one and the run of our recursive procedure is finished.

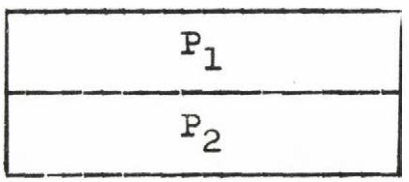
2) DO-WHILE structograms and recursive structograms

The set of DO-WHILE structograms is defined by recursion (cf. [6])

a) there is a set of elementary commands (empty, assigment...),

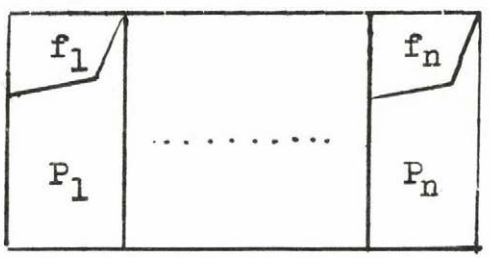
b) If P_1, P_2, ..., P_n, CM are elementary commands or already defined DO-WHILE structograms, then the following constructions are DO-WHILE too (cf. [6]).

1)

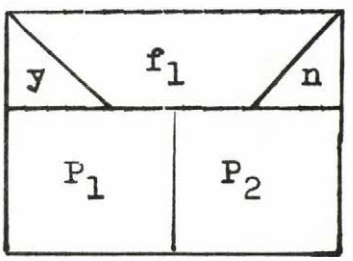


sequence construction

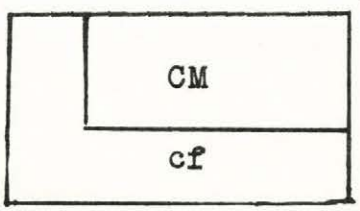
2)



or



IF constructions



DO-WHILE nem szükséges

DO-WHILE construction

With the aim of decreasing the extent the complexity of our diagrams, we write their procedure name instead of their

physical presence. Using this concept of procedures it is clear, that in the case of DO-WHILE structograms the recursive calls are prohibited.

To each structogram belongs a declaration part written in the top the structogram (in the declaration head). It contains 1) the name and the list of formal parameters

2) the list of global variables, explicitly occuring on the page
3) the list of local variables, explicitly occuring on the page
4) the list of formal parameters hand led by value (they are used to give beginning value)
5) the list of formal parameters handled by result (which are used to return the results).

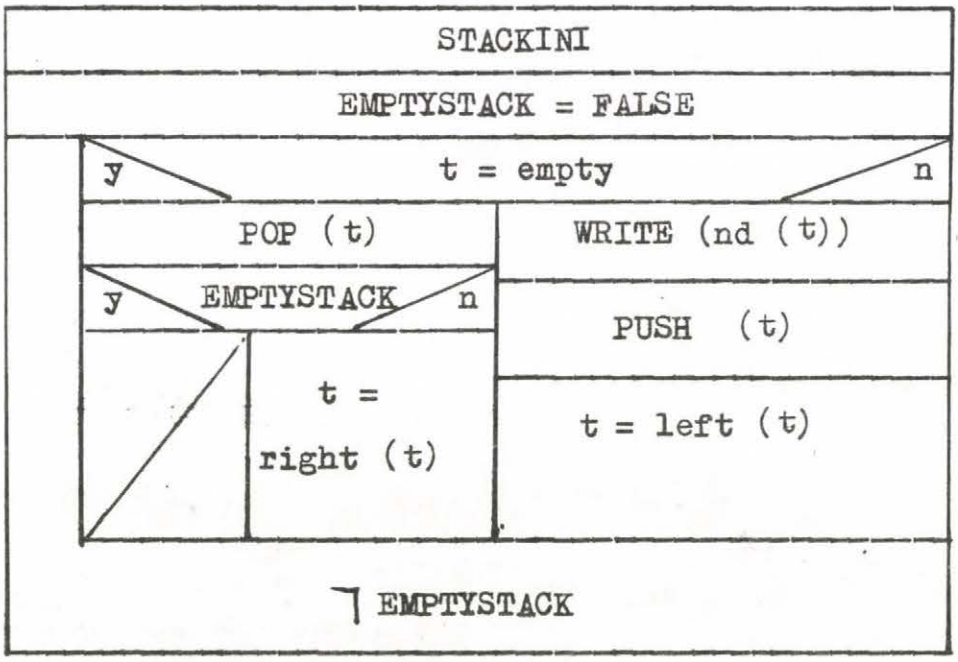
(The lists 4 and 5 have no common elemants and other forms of parameter passing are not allowed.)

Example: The iterative variant of the preorder tree traversal strategy (cf. [7]).

PRE (t)

Global: EMPTYSTACK

Value: t



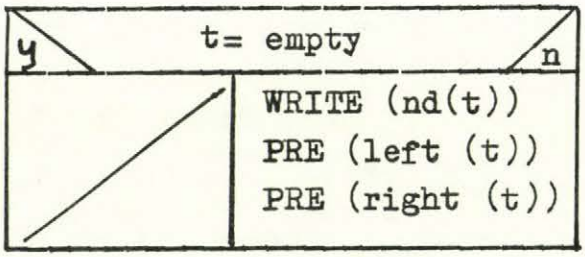
Here t is a variable of type 'binary tree', nd(t) is the root of t, left(t) and right(t) are the left and right substree of t respectively PUSH and POP are the usual stack-handling procedures and WRITE is some procedure, processing the nodes.

If instead of DO-WHILE control-structure the simple recursion (the procedure may call itself directly) is allowed we get the concept of recursive structograms.

Example: The recursive variant of preorder tree traversal:

PRE (t)

Value: t



This program is well structured, simple and obviously correct. The same cannot be said about the iterative variant.

3) The recursion removal algorithm

Let our recursive structogram the following:

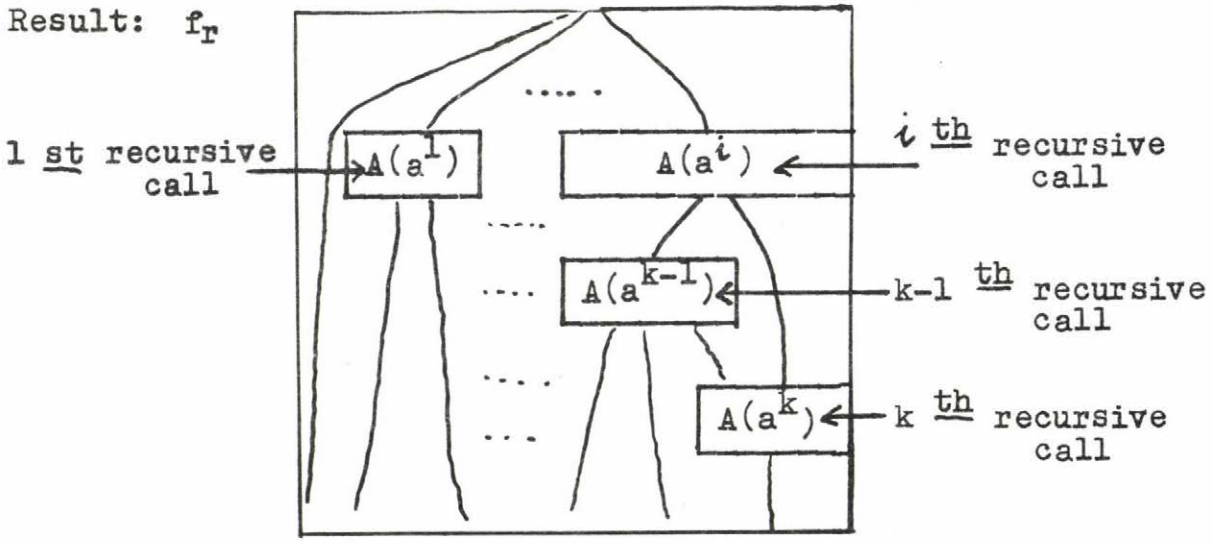
▲ (f)

Local: l

Global: g

Value: f_v

Result: f_r



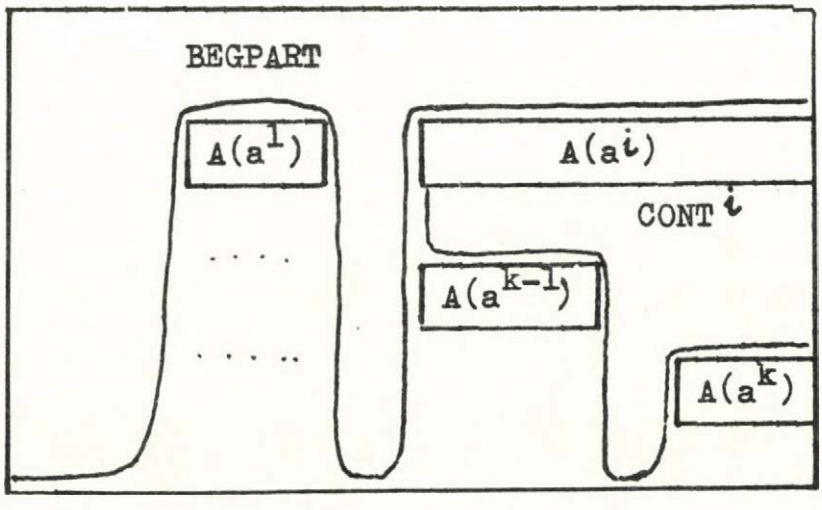
(where f, ℓ, g, f_v, f_r are, in vector form, the formal parameters, local variables, global variables, formal parameters passing by value and by result respectively. We number the recursive calls in some way. The vector of actual parameters of the call number is noted by a^i, a^i_v and a^i_r are the vectors of actual parameters handled by value and by result respectively.)

Let us cut A into pieces wich do not contain recursive calls.

We do it in the following way.

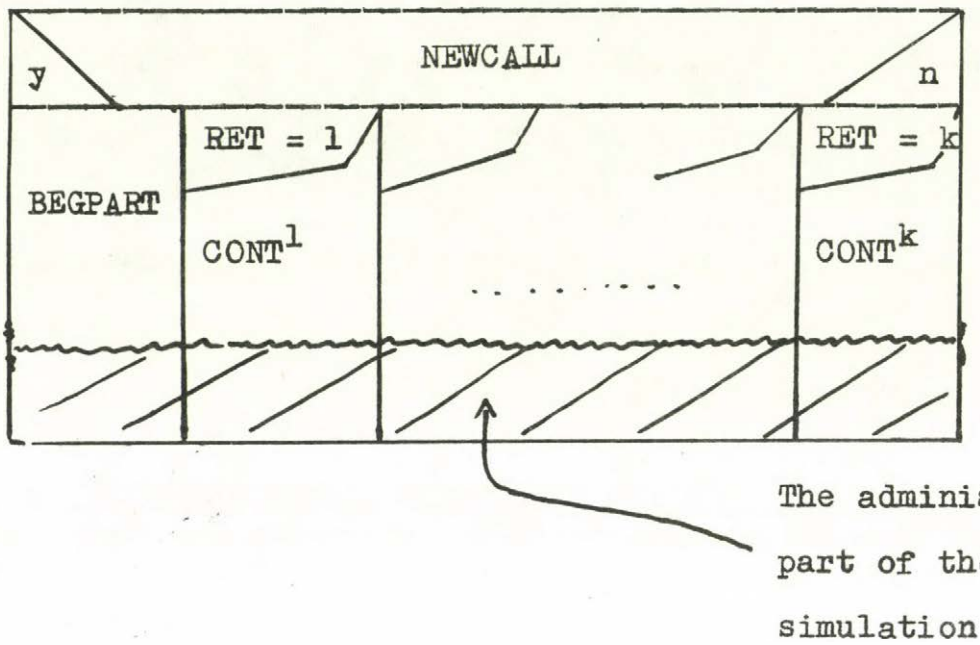
BEGPART cousists of those program branches of A which pass from the top to some recursive call (these ends are called type TRUE) or the bottom of A (ends of type FALSE)

$CONT^i$ consists of those program brauches of A which pass from the recursive call of number i to another recursive call (end of type TRUE) or the bottom of A (ends of type FALSE)



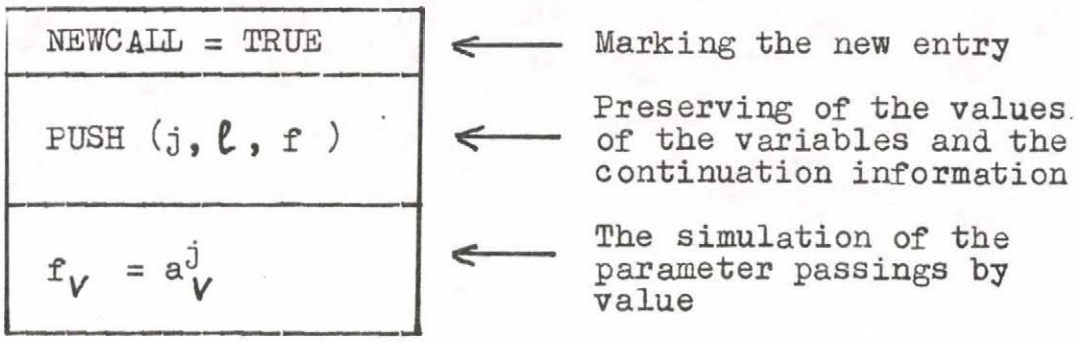
The kernel of the recursion-simulating loop has the following form.

KERN

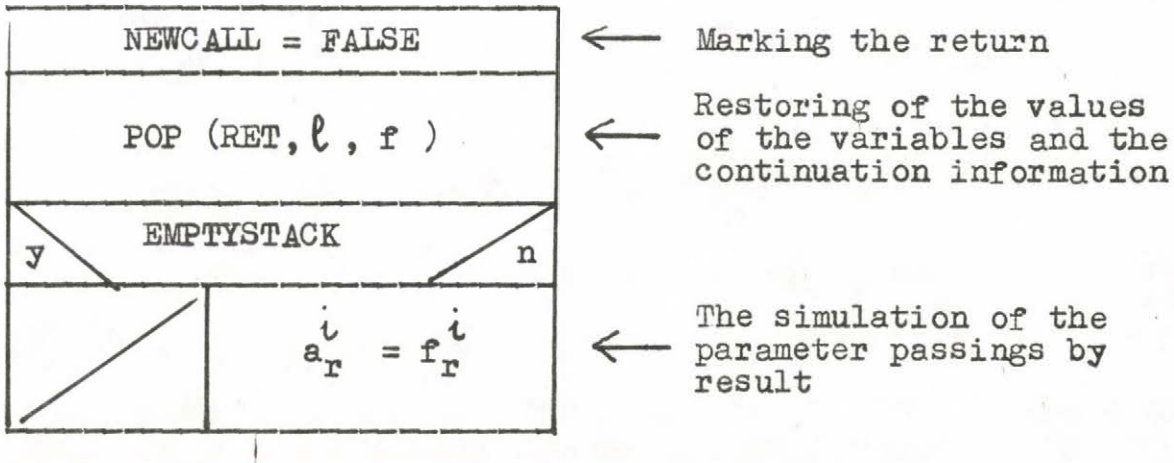


(NEWCALL is a new logical variable and RET is a new integer variable.)

The simulation activity in the administrative part has two hinds, depending on whether we must begin a new recusive level (ends of type TRUE) or we must return to an earlier level (ends of type FALSE). In the firs case we complete the branch with the following commands (suppose that the branch is finished at the recursive call of number j):

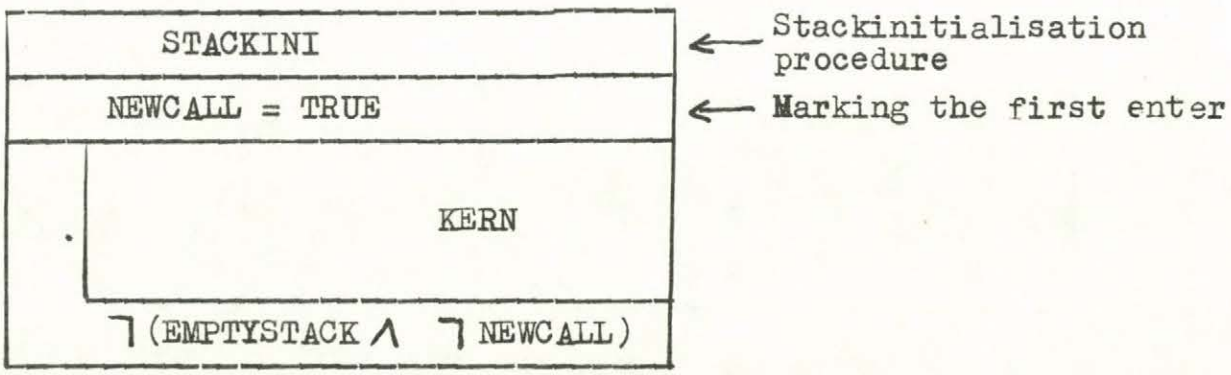


In the second case the administrative part is the following.

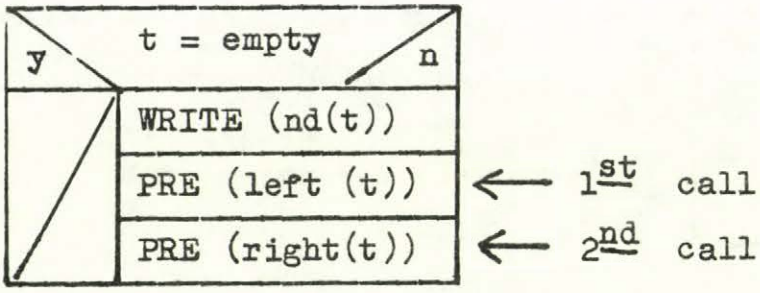


(EMPTYSTACK is a logical variable, being TRUE iff the stack is empty) Here NEWCALL = FALSE and EMPTYSTACK = TRUE means that the topmost level is finished, that is we must come out from the loop.

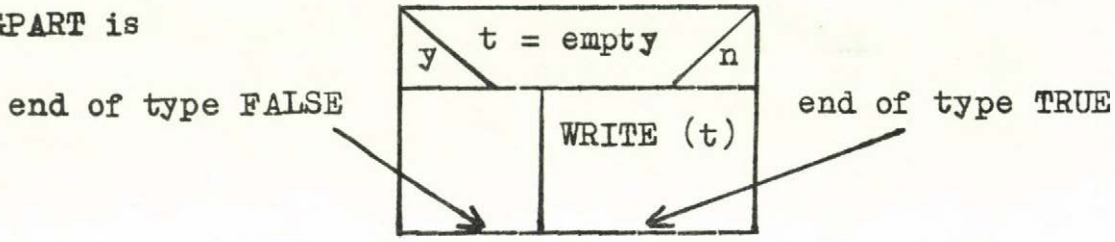
So the simulation loop has the form



Now apply our method to the procedure PRE(t)



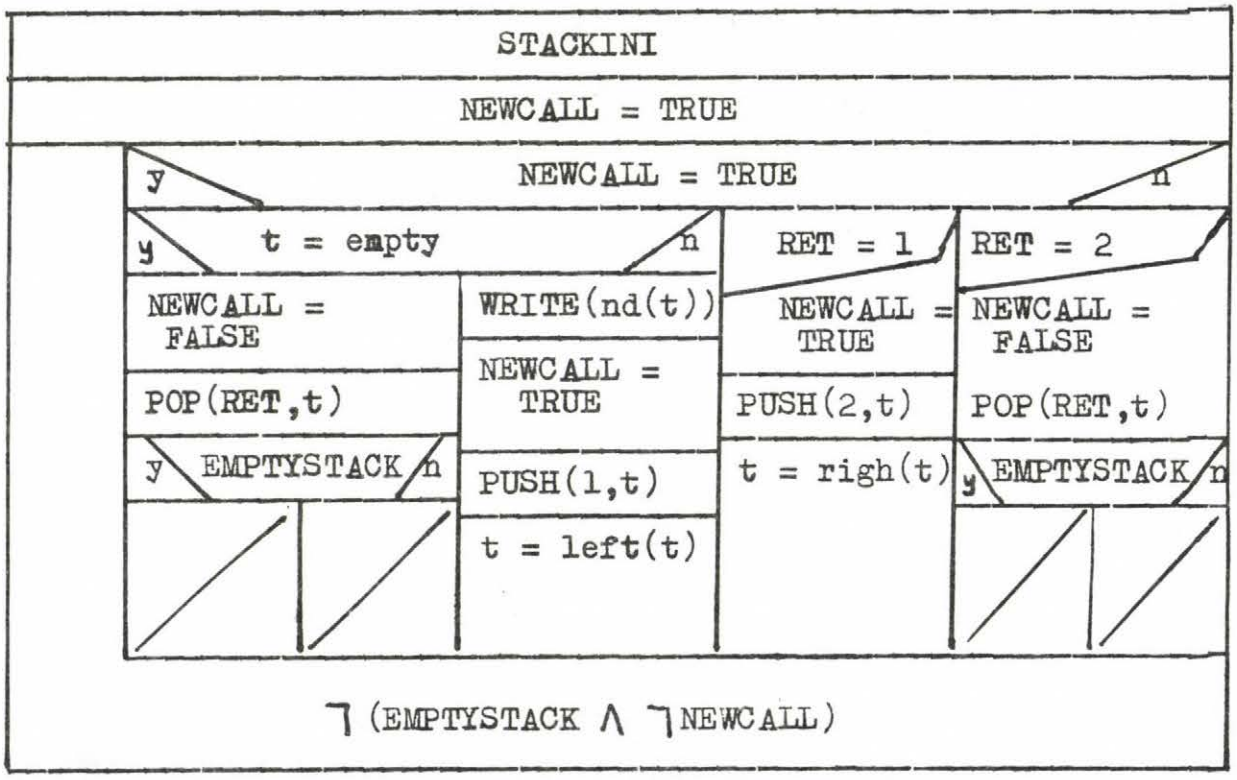
BEGPART is



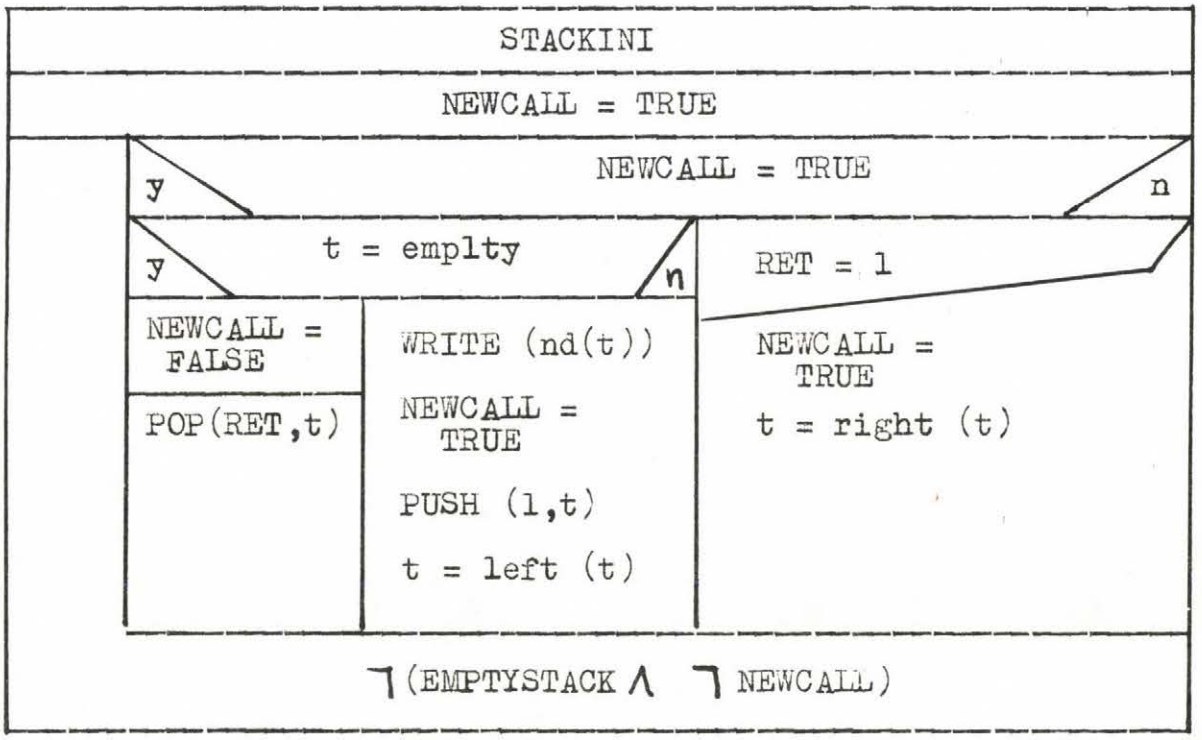
$CONT^1$ is empty with the end of type TRUE and

$CONT^2$ is empty too with the end of type FALSE.

The iterative variant is the following:



If we observe, that in the case RET = 2 the restored information is not used (it is immediatly ereased by POP (RET,t)) we can left the command PUSH (2,t) and the whole branch belonging to RET = 2 we get an equivalent program:



It is clear that to store the continuation information in RET is unnesessary here. It is clear too, that the only action t = right (t) at the FALSE value of NEWCALL can be done at the end of branch t = empty, if the EMPTYSTACK is FALSE. So the variable NEWCALL can be left, if we initialise the EMPTYSTACK to FALSE. The modified algorithm agrees with the earlier mentioned iterative variant of the preorder tree traversal.

REFERENCES

[1] H.R.Strong; Translating Recursion Equations into Flow Charts, JCSS, Vol. 5, No 3, 254-284.

[2] S.A.Walker, H.R.Strong; Characterizations of Flowchartable Recursions, JCSS, Vol. 7, No 4, 404-446.

[3] H.Partsch, P.Pepper; A family of Rules for Recursion Removal, Inf. Proc. Lett., Vol. 5, No 6, 174-177.

[4] R.S.Bird; Notes on Recursion Eliminations, CACM, Vol. 20, No 6, 434-439.

[5] N.Chapin; New Format for Flowcharts, Software Practice and Experience, Vol. 4, 341-357.

[6] E.W.Dijkstra; A Discipline in Programming, Prentice-Hall, 1976.

[7] D.E.Knuth: The Art of computer programming. Addison-Wesley Pub. Comp., 1968. .

ON THE SEMANTIC AND ECONOMIC OPTIMALITY OF PROGRAMS

by
Antal Iványi
(Eötvös University of Budapest)

Introduction

Let us consider the following task. It is given the specification of a program and we need the output of the given program, for example every month in 5 years. Let us suppose, we know different programming methods, we have different computers, and every computer has good software, including different compilers with code optimization possibilities.

A usual method to solve this task is the following one. We write a program on high level language, we debug and test it (or we try to prove its correctness), and so we get the so called source program. After this we compile the source program and get a program in binary code, the so called object program. At last we run the object program and get the necessary output, the wished results.

Fig. 1 shows this typical process: at first we have the specification, at last we need the results, and the intermediate stations are the source program and the object program.

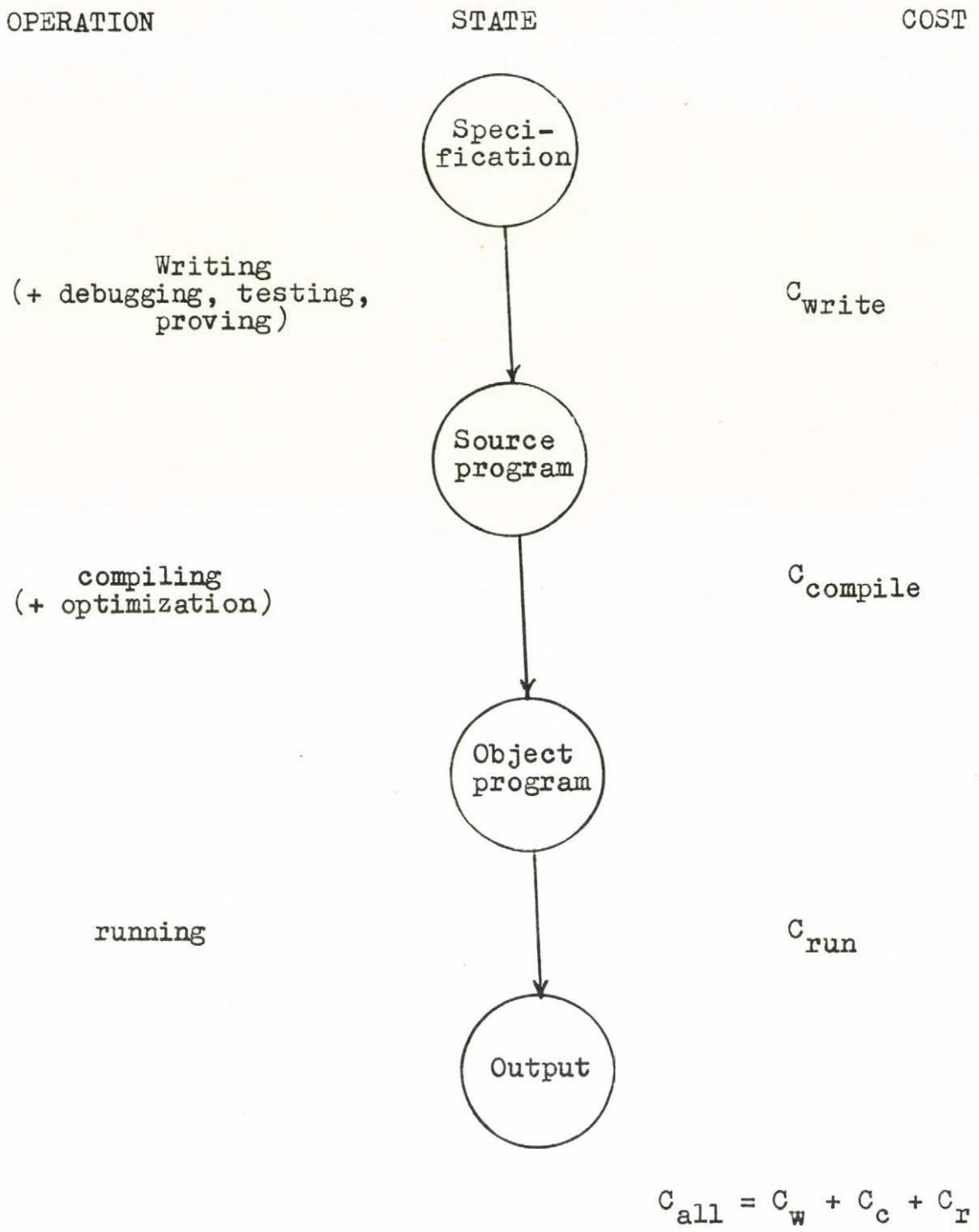


Figure 1. Typical programming process

The operations, which we have to execute, are writing, compiling, and running of programs. Of course, every operation has its own cost. We denote the corresponding costs by C_{write}, $C_{compile}$ and C_{run}. For a user, who has to pay for the program-output, a reasonable purpose is to minimize the sum of this costs.

According to the present state of computer practice there is no a global, general method for program writing, compiling and running, which can guarantee the minimal all cost. Even there is no a unique method to compute the concrete cost for a given operation.

The actions of this process today are investigated independently. For example, when professor Kobayashi enumerated the most important performance measures-turnaround time, throughput, response time, resource utilization-he considered the running performance. Professor Goldsack was dealing with specification problems.

The present state of computer practice was characterized last december on the Neumann-congress by Mr. Bedő as follows. There is a large gap between the programming an running methodology: the easily modifyable, easily usable, readable programs are not efficient - on the other side, to modify an efficient program - almost impossible.

This paper gives a report about the beginning steps of a research work, which is made by Computer Science Group of Eötvös University, and is sponsored by Research Institute

for Computer Applications, or Hungarian - by SZÁMKI.

The aim of our work is to decrease the gap between the programming and running methods, and in such a way to decrease the sum of the costs.

One of our concrete purposes is to define the semantically optimal programs.

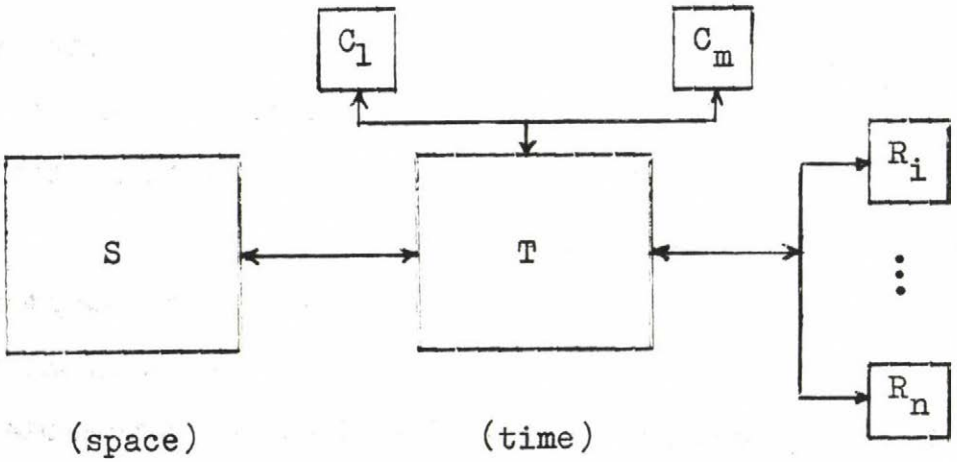
The basic idea was proposed by Aho and Ullman: taking into account the space and time reguirements, to define the running cost of programs, and to choose from the set of equivalent programs - as semantically optimal one - the program with minimal running cost.

The sketch of this paper is as follows.

The first part is a simple computer model, the second one contains the necessary definitions, the third part contains some properties of the defined concepts, and finally - some remarks.

1. Scheme of a computer

We use the following scheme of computers. (Fig. 2.)



S main memory (q bits)

T processor

C_i (i = 1,...,m) compilers

R_i (i = 1,...,n) resources (with capacity r_i bits)

Figure 2. Scheme of a computer

About the object programs (in the following shortly: programs) is supposed:

a) their correctness is proved;

b) their input data (x) are given as an ordered sequence of bits, stored on the resources R_i, with maximal length $\sum_{i=1}^{n} r_i$;

c) their commands are given as an ordered sequence of bits stored in S, with maximal length q;

d) the running of programs ends in a finite time, and the output data (y) appear on the resources R_i with maximal length $\sum_{i=1}^{n} r_i$.

2. Definitions

We use the following definitions.

Def. 1. A computer (M) is an (m+n+2)-tuple of positive real numbers $M = (\sigma, \tau, \beta_1, \ldots, \beta_m, \rho_1, \ldots, \rho_n)$, where

m the number of compilers;

n the number of resources;

σ constant (Ft/sec), the cost of the use of 1 bit (in S) for 1 sec;

τ constant (Ft/sec), the cost of the use of T for 1 sec;

β_i (i = 1, ..., m) constant (Ft/sec), the cost of the use of C_i for 1 sec;

ρ_i (i = 1, ..., n) constant (Ft/sec), the cost of the use of R_i for 1 sec.

Def. 2. A program (P), belonging to a given computer M, is an (n+4)-tuple of positive real numbers $P = (p, t, j, t_c, t_1, \ldots, t_n)$, where

p the length of P in bits ($p \leq q$);

t the running time of P, sec (t is finite);

j the index of compiler, used to produce P;

t_c the time (sec), used to produce P

t_i (i = 1, ..., n) the time (sec) of the use of i-th resource during the running of P;

Def. 3. Programs P_1 and P_2 are equivalent, iff their input data and their output data are the same, i.e.

$$x_1 = x_2 \text{ and } y_1 = y_2.$$

Def. 4. The running cost (C_{run}) of a program P for a computer M is

$$C_{run}(P,M) \underline{\text{def}} \sigma \cdot s + \tau \cdot t + \sum_{i=1}^{n} \rho_i t_i.$$

Def. 5. All cost (C_{all}) of a program P for a computer M, for k running is

$$C_{all}(P,M,k) = t_c \cdot \beta_j + k \cdot C_{run}.$$

Def. 6. Let $\mathcal{T}$ a set of equivalent programs, μ a given set of computers. We say that $P \in \mathcal{T}$ is semantically optimal for $\mathcal{T}$ and μ iff

$\forall P' \in \mathcal{H}$, $\forall M' \in \mu$ $\quad C_{run}(P',M') \geq C_{run}(P,M)$.

Def. 7. Let $\mathcal{T}$ a set of equivalent programs, μ a given set of computers. We say that $P \in \mathcal{T}$ for k runs is economically optimal for $\mathcal{T}$, and μ iff

$\forall P' \in \mathcal{T}$, $\forall M' \in \mu$ $\quad C_{all}(P',M',k) \geq C_{all}(P,M,k)$.

3. Features of the defined concepts

The following two assertions give some simple features of the defined concepts.

Ass. 1. Let $\mathcal{T}$ be a set of equivalent programs, and let μ be the set of computers, belonging to the elements of $\mathcal{T}$. Then there exists a syntactically optimal program for μ and $\mathcal{T}$.

Ass. 2. Let $\mathcal{T}$ be a set of equivalent programs, and let μ be the set of computers, belonging to the elements of $\mathcal{T}$. Then for any given number of runnings k there exists an economically optimal program for μ and $\mathcal{T}$.

4. Remarks

Finally we have several remarks. This remarks show some disadventages of the proposed definitions, on the other side these remarks show the development posibilities.

a) The given approach considers only the running

characteristics of programs. There are other semantic features too, for example portability.

b) This model does not take into account some features of multiprogrammed systems (for example, the parameters can depend on the environment).

c) The model does not take into account the possibility of system failures.

d) The running time, required space and necessary peripheries of real programs depend on input data, therefore in a more precise model we have to use expected values instead of constants.

e) Because the set of equivalent programs is finit under the given assumptions, generating all elements, computing and comparing their costs we have a constructive way to produce the semantically and the economically optimal programs.

Of course, for the practice we need other, more effective methods.

5. Further plans

a) We would like: to give a method to determine the cost of program writing. After this we have a quantitative method to compare the different methods;

b) to collect the mentioned constants for different computers;

c) to estimate the minimal, theoretically necessary

costs for concrete applications, and to compare the minimal and real costs.

d) Not is the next future we would like to connect the programming and running methods - and to find a good compromiss between these activities.

REMARKS ON THE STRUCTURE OF UNMARKED PETRI NETS

Ryszard Janicki

Institute of Mathematics, Warsaw Technical University,

Pl. Jedności Robotniczej 1, 00-661 Warszawa, Poland.

Introduction.

Petri nets ([1,2,3,4]) have been used to represent the behaviour of concurrent systems. In Petri's model, unmarked nets represent static aspects of dynamic systems, whereas marked nets represent their dynamic aspects. It turns out that much dynamic aspects of nets can be described by means of static aspects, that is to say the static structure of nets describes all potentially possible dynamic properties.

In this paper we shall only deal with properties of unmarked nets. It turns out that these nets create a lattice, sequential nets (called elementary in the paper) are atoms, and nets generated by elementary nets have properties required from well-defined dynamic systems (safeness, reachability and so on). The notation used in the paper originates from [1].

Simple nets.

In this section we recall from [1] the basic notion of this approach, namely the notion of simple net.

For every set X, let $left: X \times X \rightarrow X$, $right: X \times X \rightarrow X$ be the following functions:

$(\forall (x,y) \in X \times X)$ $left((x,y))=x$, $right((x,y))=y$.

By a simple net (abbr. s-net) we mean any pair:

$$N = (T,P),$$

where: T is a set (of transitions),

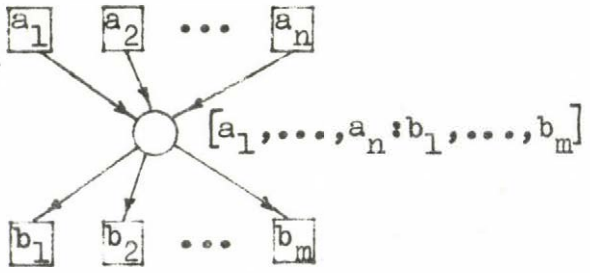
$P \subseteq 2^T \times 2^T$ is a relation (interpreted also as a set of places),

$(\forall a \in T)(\exists p,q \in P) \quad a \in left(p) \cap right(q)$,

$P=\emptyset \Leftrightarrow T=\emptyset$.

In the paper we restrict our consideration to finite simple nets. Instead of $(\{a_1,\ldots,a_n\},\{b_1,\ldots,b_m\})$ we shall write $[a_1,\ldots,a_n:b_1,\ldots,b_m]$.

Every s-net N=(T,P) can be represented graphically, namely every place $[a_1,\ldots,a_n:b_1,\ldots,b_m]$ can be represented by the following graph:

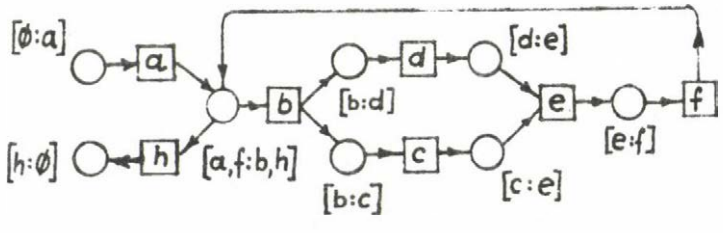


Example 1.

Let N=(T,P), where $T=\{a,b,c,d,e,f,h\}$,

$P = \{[\emptyset:a],[a,f:b,h],[h:\emptyset],[c:e],[b:c],[b:d],[d:e],[e:f]\}$.

The pair N=(T,P) is a s-net and it can be represented by the following graph.



■

In the literature nets are usually defined differently, starting with two disjoint sets of transitions and places, and introducing a flow-relation among them (see Petri[4]). This approach is luckier in the sense that it makes more easy to handle operations among nets.

Of course, not every Petri net defined in the general way (i.e. according to [4]) is a simple net. Simple nets are such kind of Petri nets defined generally that every place is unambiguously defined by entrances and exits of transitions.

A lattice of simple nets.

We shall now describe the algebraic structure of simple nets.

Let SNETS denote the family of all finite s-nets. Note that SNETS is a set.

Let $\sqsubseteq$ be the relation in SNETS defined as follows:

$$N_1=(T_1,P_1) \sqsubseteq N_2=(T_2,P_2) \iff P_1 \subseteq P_2 .$$

Note that $\sqsubseteq$ is a partial order relation and $N_1 \sqsubseteq N_2 \Rightarrow T_1 \subseteq T_2$

Let $\sup\{N_1,N_2\}$, $\inf\{N_1,N_2\}$ denote respectively the least upper bound and the greatest lower bound with respect to the relation $\sqsubseteq$.

Lemma 1.

$(\forall P_1,P_2 \subseteq 2^T \times 2^T)$ $\quad left(P_1)=right(P_1)$ & $left(P_2)=right(P_2) \Rightarrow$

$\Rightarrow \quad left(P_1 \cup P_2)=right(P_1 \cup P_2)$

Proof.

$a \in left(P_1 \cup P_2) \iff (\exists\, p \in P_1 \cup P_2)\ a \in left(p) \iff (\exists\, p \in P_1)\ a \in left(p)$ or

$(\exists\, p\in P_2)\ a\in left(p) \iff a\in left(P_1)$ or $a\in left(P_2) \iff$

$\iff a\in left(P_1)\cup left(P_2)$.

In the same way we prove that $right(P_1\cup P_2)=right(P_1)\cup right(P_2)$, but this ends the proof. ■

Theorem 2.

For every $N_1=(T_1,P_1)$, $N_2=(T_2,P_2)\in$ SNETS:

$\sup\{N_1,N_2\} = (T_1\cup T_2, P_1\cup P_2)$,

$\inf\{N_1,N_2\} = (left(P),P)$, where P is the greatest set such that: $P\subseteq P_1\cap P_2$ & $left(P)=right(P)$.

Proof.

Let $N=(T,P)=\sup\{N_1,N_2\}$. From Lemma 1 we obtain that the pair $(T_1\cup T_2, P_1\cup P_2)$ is a s-net. Of course, $N_i\sqsubseteq (T_1\cup T_2, P_1\cup P_2)$ for i=1,2. From the definition of $\sqsubseteq$ we have that $T_1\subseteq T$, $T_2\subseteq T$, $P_1\subseteq P$, $P_2\subseteq P$. This fact means that $(T_1\cup T_2, P_1\cup P_2)\sqsubseteq N$.

But N is the least upper bound then $(T_1\cup T_2, P_1\cup P_2)=N$.

The second fact follows directly from the definition. ■

Define the following operations:

$$N_1\cup N_2 = \sup\{N_1,N_2\},\quad N_1\cap N_2 = \inf\{N_1,N_2\},$$

$$\bigcup_{N\in S} N = \sup\{N\mid N\in S\},\quad \bigcap_{N\in S} N = \inf\{N\mid N\in S\}.$$

Theorem 3.

The algebra (SNETS,$\cup$,$\cap$) is a lattice with the greatest lower bound $(\emptyset,\emptyset)$.

Proof.

This follows from Theorem 2 and the definition of operations $\cup$, $\cap$. ■

It turns out that the lattice (SNETS,$\cup$,$\cap$) is not distributive.

We are now going to discuss the atomic structure of simple nets.

A s-net N=(T,P) is said to be an <u>atom</u> iff :

$N \neq (\emptyset,\emptyset)$ & $\left[(N' \sqsubseteq N) \Longrightarrow (N'=N \text{ or } N'=(\emptyset,\emptyset)\right]$.

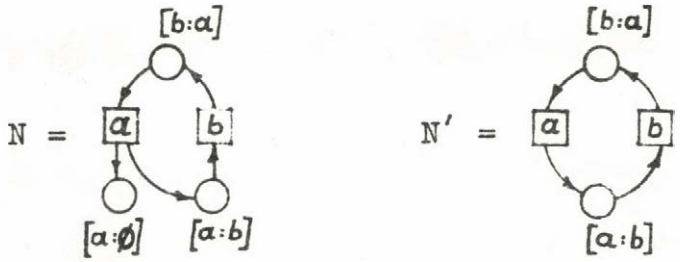
Other words, the net N is the atom if it is an atom of the lattice (SNETS,$\cup$,$\cap$). For every s-net N, let atoms(N) denotes the set of all atoms which were contained in N, i.e.

$$\text{atoms}(N) = \{N' \mid N' \sqsubseteq N \ \&\ N' \text{ is an atom}\} .$$

A s-net N is said to be <u>atomic</u> iff $N = \bigcup_{N' \in \text{atoms}(N)} N'$.

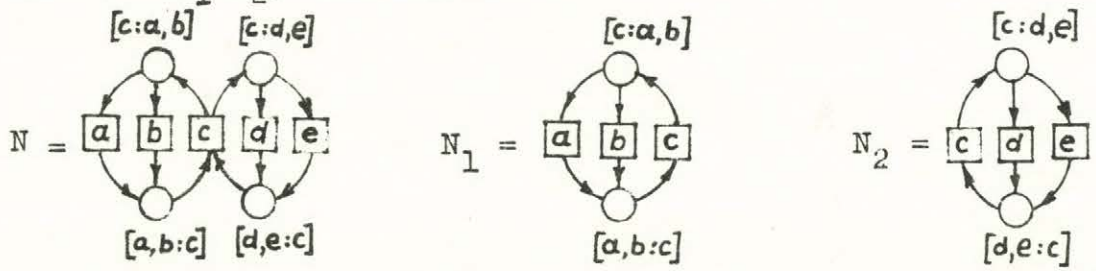
<u>Example 2.</u>

Let N, N' be nets defined below. Note that atoms(N)=$\{N'\}$ and $N' \neq N$, so the net N is not atomic.



<u>Example 3.</u>

Let N, N_1, N_2 be nets defined below. In this case atoms(N)=$\{N_1, N_2\}$ and $N = N_1 \cup N_2$, thus the net N is atomic.



■

Let $N=(T,P)$ be a simple net. To simplify the considerations, we shall use the following well known notation:

1. $(\forall p\in P)\quad p^\bullet = right(p),\ {}^\bullet p = left(p)$,
2. $(\forall a\in T)\quad a^\bullet = \{p\in P \mid a\in left(p)\}$, ${}^\bullet a = \{p\in P \mid a\in right(p)\}$.

If the net N is not understood, we shall write $({}^\bullet a)_N$, $(a^\bullet)_N$, $({}^\bullet p)_N$, $(p^\bullet)_N$. Of course, $(\forall p\in P)\quad p=({}^\bullet p, p^\bullet)$.

Note that the above operations are correctly defined for every pair (T,P), where $P\subseteq 2^T\times 2^T$.

Lemma 4.

A pair (T,P), where $P\subseteq 2^T\times 2^T$ is a simple net iff:

$$(\forall a\in T)\quad a^\bullet\neq\emptyset \ \&\ {}^\bullet a\neq\emptyset.$$

Proof.

In order to prove the above lemma it is enough to show that:

$[(\forall a\in T)\ a^\bullet\neq\emptyset]\Longleftrightarrow left(P)=T$, and $[(\forall a\in T)\ {}^\bullet a\neq\emptyset]\Longleftrightarrow right(P)=T$.

And so: $[(\exists a\in T)\ a^\bullet=\emptyset]\Rightarrow[(\exists a\in T)\ \{p\in P \mid a\in left(p)\}=\emptyset]\Rightarrow$

$[(\exists a\in T)(\forall p\in P)\ a\in left(p)]\Rightarrow[(\exists a\in T)\ a\in left(P)]\Rightarrow\ left(P)\neq T$.

Thus: $(\forall a\in T)\ a^\bullet\neq\emptyset \Leftarrow\ left(P)=T$.

On the other hand:

$[(\forall a\in T)\quad a^\bullet=\{p\in P \mid a\in left(p)\}\neq\emptyset]\Rightarrow[(\forall a\in T)(\exists p\in P)\ a\in left(p)]\Rightarrow$

$\Rightarrow\ T\subseteq left(P)$.

From the definition we have $left(P)\subseteq T$, then

$[(\forall a\in T)\quad a^\bullet\neq\emptyset]\Rightarrow left(P)=T$.

To prove the next point we proceed analoguously. ■

Let $F\subseteq T\times P\cup P\times T$ (or F_N, if N is not understood) be the following relation:

$(\forall x,y\in P\cup T)\quad (x,y)\in F\ \Leftrightarrow\ x\in left(y)$ or $y\in right(x)$.

Note that for every s-net (T,P), the triple (T,P,F) is Petri's definition of net.

A s-net $N=(T,P)$ is said to be <u>connected</u> iff:

$$(\forall x,y\in T\cup P)\quad (x,y)\in (F\cup F^{-1})^{*}\ .$$

Other words, the net N is connected if the direct graph $(T\cup P,F)$ is connected.

<u>Theorem 5</u>.

Every atom is connected.

<u>Proof.</u>

We shall prove that if s-net is not connected then it is not an atom. Define $C_T\subseteq T\times T$ in the following way:

$$C_T = (F\cup F^{-1})^{*}\cap T\times T.$$

Let $N=(T,P)$ be a net. Assume that N is not connected.

This means that $T\times T - C_T \neq \emptyset$.

Note that C_T is an equivelence relation. For every $a\in T$, let $[a]_{C_T}$ denote the equivalence class of C_T containing the element a.

We set $T_a=[a]_{C_T}$. Since $T\times T - C_T \neq \emptyset$ then $T_a\subsetneq T$.

Of course, $a\in T_a$. Let $P_a=\{p\in P \mid (\exists b\in T_a)\ \ b\in {}^{\bullet}p\cup p^{\bullet}\}$.

Because $((\forall b\in T-T_a)\ \ b\in {}^{\bullet}p\cup p^{\bullet})\Rightarrow p\notin P_a$ then $P_a\subsetneq P$.

We are going to prove that (T_a,P_a) is a simple net. To this end it is enough to prove that: $left(P_a)=right(P_a)=T_a$.

Let $b\in left(P_a)$. This means that $(\exists p\in P_a)\ b\in {}^{\bullet}p$.

Since $p\in P_a$, then from the definition of P_a we obtain:

$(\exists b'\in T_a)\ \ b'\in {}^{\bullet}p\cup p^{\bullet}$.

Thus, $\{b,b'\}\subseteq {}^{\bullet}p\cup p^{\bullet}$, so $(b,b')\in (F\cup F^{-1})^{2}\cap T\times T\subseteq C_T$.

But this means that $b\in [b']_{C_T}=[a]_{C_T}=T_a$.

Hence $left(P_a)\subseteq T_a$.

In the analoguous way we prove that $right(P_a)\subseteq T_a$.

Let $b\in T_a$. Note that: $(\forall p\in b^{\bullet})\ \ b\in {}^{\bullet}p\subseteq {}^{\bullet}p\cup p^{\bullet}$, then $b^{\bullet}\subseteq P_a$.

Let $p\in b^{\bullet}$. Then $b\in {}^{\bullet}p\subseteq left(P_a)$, therefore $T_a\subseteq left(P_a)$.

For T_a and $right(P_a)$ we proceed analoguously.

Thus: $(\forall a \in T)\ (T_a, P_a) \not\subseteq (T,P)$, then (T,P) is not an atom. ■

Elementary nets.

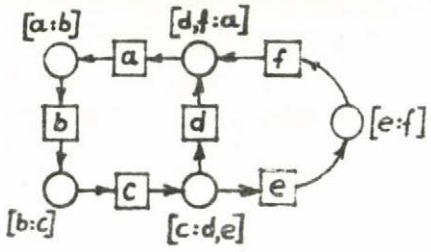
In this section we define s-nets representing sequential systems and processes. We shall prove that these nets are special kind of atoms.

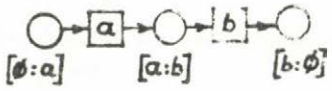
A simple net $N=(T,P)$ is said to be elementary iff

1. $(\forall a \in T)\ \mathrm{card}({}^\bullet a) = \mathrm{card}(a^\bullet) = 1$,
2. N is connected.

Example 4.

Two nets defined below are elementary.





■

Theorem 6.

Every elementary net is an atom.

Proof.

Suppose that $N=(T,P)$ is an elementary net, and there is a net $N' = (T',P')$ such that: $(\emptyset,\emptyset) \neq (T', P') \not\subseteq (T,P)$.

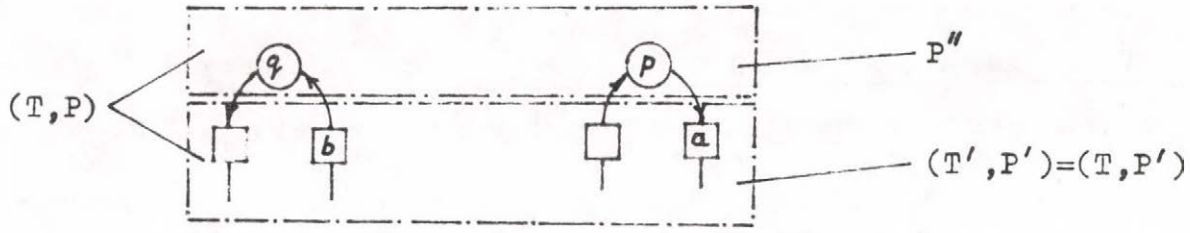
We must consider two disjoint cases, namely:

1. $T' = T$ & $P' \not\subseteq P$.
2. $T' \not\subseteq T$ (of course, $T' \not\subseteq T \Rightarrow P' \not\subseteq P$).

Case 1.

Let $P'' = P - P'$.

Then the net (T,P) is of the following form:



Since it is the elementary net then $(\forall a\in T)\ card(a^\bullet)=card(^\bullet a)=1$.

Let $a\in T'-T$ such that $^\bullet a=\{p\}\subseteq P''$. This means that $a\notin right(P')$, so (T,P') is not a simple net.

Let $b\in T'$ such that $b^\bullet=\{q\}\subseteq P''$. This means that $b\notin left(P')$.

Since $P''\neq\emptyset$ then there is such $a\in T$ that:

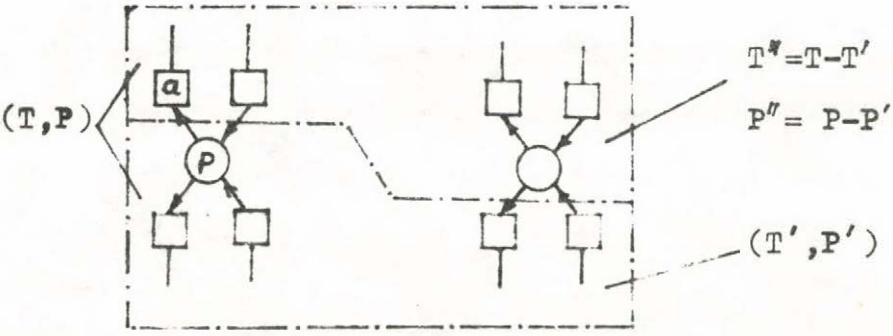
$$(^\bullet a\cap P'')\cup(a^\bullet\cap P'')\neq\emptyset.$$

Thus, the assumption $P''=P-P'\neq\emptyset$ leads to the discrepancy.

Case 2.

Here $T'\subsetneq T$, $P'\subsetneq P$.

Let $P''=P-P'$, $T''=T-T'$. This case can be schematically drawed in the following way:



The net (T,P) is connected because it is elementary.

Then there is $a\in T''$ such that $(^\bullet a\cup a^\bullet)\cap P'=\emptyset$.

Other words: $(\exists a\in T'')(\exists p\in P')\ \ p\in{}^\bullet a\cup a^\bullet$.

Suppose that $p\in{}^\bullet a$. Since it is the elementary net then $^\bullet a=\{p\}$.

But this means that: $a\in right(p)\subseteq right(P')$ and $a\notin T'$.

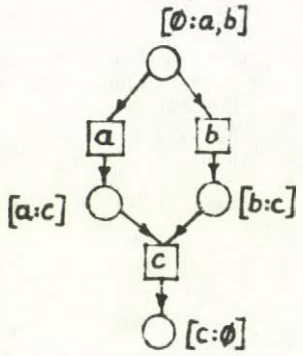
Hence $right(P')\neq T'$.

Analoguously, the assumption $p\in{}^\bullet a$ implies that $left(P')\neq T'$.

Thus the pair (T',P') is not s-net, in spite of the assumption.

Then every elementary net must be an atom. ■

Of course, not every atom is an elementary net. For example, the below s-net is an atom, but it is not elementary.



For every s-net N, let

$$elem(N) = \{N' \mid N' \sqsubseteq N \ \& \ N' \text{ is elementary}\}.$$

Of course $elem(N) \subseteq atoms(N)$, and generally this inclusion is proper one. Note that every elementary net is equivalent with a totally labelled state machine.

Proper nets.

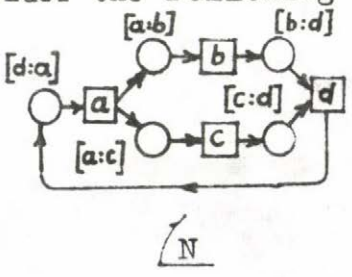
We shall now define the most important class of s-nets, namely the class of so called proper nets. We suppose that proper nets describe the static structure of "well defined" concurrent dynamic systems and processes.

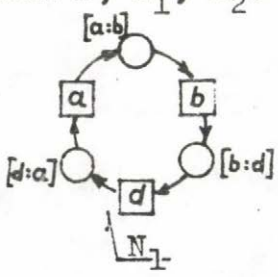
A s-net N is said to be proper iff: $N = \bigcup_{N' \in elem(N)} N'$.

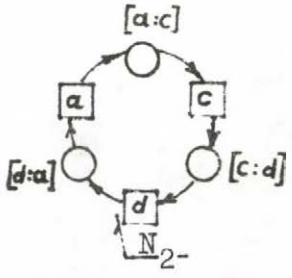
Of course, every proper net is atomic, but not every atomic net is proper.

Example 5.

Consider the following three nets N, N_1, N_2.



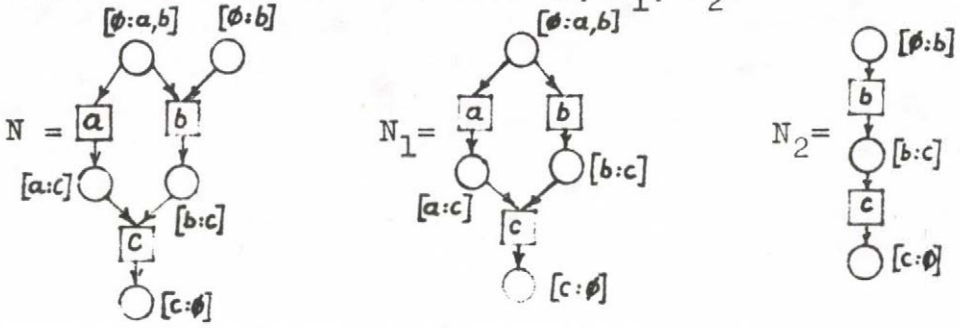




Note that $elem(N)=\{N_1,N_2\}$, $N=N_1 \cup N_2$, so the net N is proper.

Example 6.

Consider the next three nets : N, N_1, N_2.



In this case: $elem(N)=\{N_2\} \subsetneq atoms(N)=\{N_1,N_2\}$, $N \neq N_2$, then the s-net N is not proper, although it is atomic. ■

Corollary 7.

A s-net N=(T,P) is proper $\Longleftrightarrow$ there exists a set $\{N_1,\ldots,N_m\}$ of elementary nets such that $N = N_1 \cup \ldots \cup N_m$.

Proof.

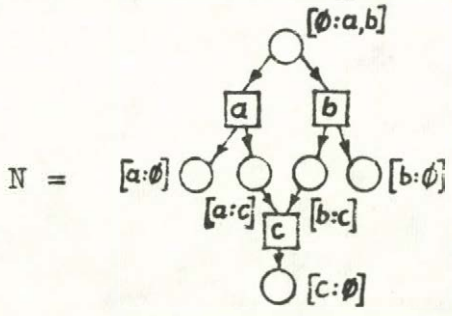
$\Rightarrow$ It is enough to set $\{N_1,\ldots,N_m\} = elem(N)$.

$\Leftarrow$ $\bigcup\{N' \mid N' \in elem(N)\} \sqsubseteq \bigcup\{N' \mid N' \in atoms(N)\} \sqsubseteq N = N_1 \cup \ldots \cup N_m$.

On the other hand $\{N_1,\ldots,N_m\} \subseteq elem(N)$.

But this means that $N = N_1 \cup \ldots \cup N_m = \bigcup\{N' \mid N' \; elem(N)\}$. ■

Consider the following s-net.



The above s-net N is elementary, but $elem(N) \neq atoms(N)$.

We leave the simple proof of this fact to the reader.

Let PNETS denote the family of all proper nets.

Lemma 8.

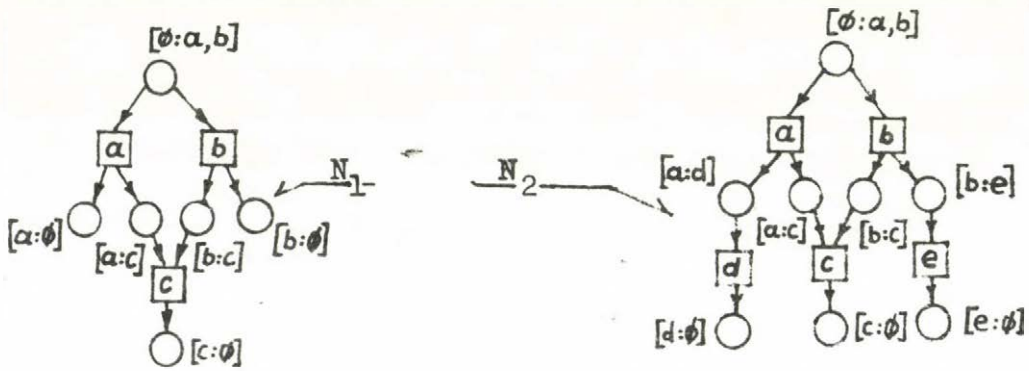
$(\forall\, N_1, N_2 \in \mathrm{PNETS})$ $\quad N_1 \cup N_2 \in \mathrm{PNETS}$.

Proof.

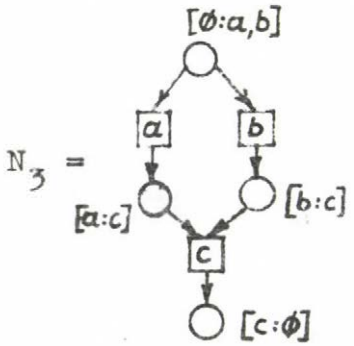
Let $\mathrm{elem}(N_1)=\{N_1^1,\ldots,N_m^1\}$, $\mathrm{elem}(N_2)=\{N_1^2,\ldots,N_n^2\}$.

$N= N_1 \cup N_2 = N_1^1 \cup \ldots \cup N_m^1 \cup N_1^2 \cup \ldots \cup N_n^2$, so from Corollary 7 we have that N is proper. ■

The family PNETS is not closed under the operation $\cap$. To prove this fact we consider the following two proper net N_1 and N_2 .



Note that $N_3=N_1 \cap N_2$ is of the following form:



and, of course, N_3 is not proper.

Proper nets are important class of s-nets. They have a few conveniant properties. Naturally marked proper nets are safe, and fireable. Proper nets play basic part in the theory of traces [1,2,3] . To author mind, they represent static structure of "well

defined" dynamic systems. Proper nets are superpositions of state machines, and this is - to author mind - the natural way of building nets representing a given concurrent systems (compare [2]).

Occurrence nets.

Nets of occurrences, introduced by Petri[4], are widely used model of concurrent processes. We shall discuss their properties formulated in terms of this paper.

A s-net N=(T,P) is said to be the occurrence net iff:

1. $(F_N)^+ \cap id = \emptyset$ (id - identity relation),
2. $(\forall p \in P)$ $card(^\bullet p) \leq 1$, $card(p^\bullet) \leq 1$.

Note that in the case of occurrence nets (see[4]) the relation $(F_N)^*$ is a partial order relation.

Theorem 9.

Every occurrence net is proper.

Proof.

Let N=(T,P) be the occurrence net.

Define $F_P = F^2 \cap P \times P$. Note that F_P^* is also a partial order relation. Let MCH be the set of all maximal chains described by the relation F_P^*. We shall prove that:

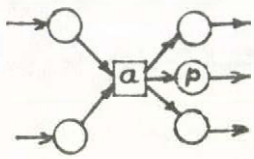
$(\forall P' \in MCH)$ (left(P'),P') is an elementary net.

At first we prove that the pair (left(P'),P) is a s-net.

To this end it is enough to prove that left(P') = right(P').

Since for every $p \in P$: $card(^\bullet p) \leq 1$, $card(p^\bullet) \leq 1$, we have that every $p \in P$ is one of the following forms: [a:b],[∅:b],[a:∅], where $a,b \in T$.

Let $a \in$ left(P').



This means that there is $p \in P'$ such that $\{a\} = {}^\bullet p$.

Since P' is a maximal chain described by F_P^+, there is $q' \in {}^\bullet a \cap P'$.

But this means that $(\exists q' \in P')\ \{a\} = (q')^\bullet$, so $a \in right(P')$.

Thus $left(P') \leq right(P')$.

In the similar way we prove that $right(P') \leq left(P')$.

Hence, the pair (T', P'), where $T' = left(P')$, $P' \in MCH$ is a s-net.

From the fact that $(\forall p,q \in P')\ (p,q) \in F_P^*$ or $(q,p) \in F_P^*$, it follows that this net is connected.

Now, we prove the following fact:

$$(\forall a \in T)(\forall p,q \in P)\ \ a \in p^\bullet \cap q^\bullet \Rightarrow (p,q) \notin F_P^+, \text{ and}$$

$$a \in {}^\bullet p \cap {}^\bullet q \Rightarrow (p,q) \notin F_P^+.$$

This fact we are going to prove by the discrepancy.

Suppose that $a \in T$, $a \in p^\bullet \cap q^\bullet$ and $(p,q) \in F_P^+$.

Let $p_1, \ldots, p_m$ be a sequence of elements of P such that

$p_1 = p$, $p_m = q$, $(p_i, p_{i+1}) \in F_P$ for $i = 1, \ldots, m-1$.

Since $(\forall p \in P)$ $card(p^\bullet) \leq 1$, $card({}^\bullet p) \leq 1$ then $p_1^\bullet = \{a\} = {}^\bullet p_2$.

Since $q^\bullet = \{a\}$ then $(q, p_2) \in F_P$, and we have $(q,q) \in F_P^+$ – in spite of the assumption that $(F_N)^+ \cap id = \emptyset$.

For $a \in {}^\bullet p \cap {}^\bullet q$ we proceed analogously.

From the above fact we obtain that:

$$(\forall a \in T')\ \ card({}^\bullet a \cap P') \leq 1\ ,\ card(a^\bullet \cap P') \leq 1,$$

because in the opposite case we have:

$p \neq q\ \&\ \{p,q\} \leq {}^\bullet a \cap P' \Rightarrow a \in p^\bullet \cap q^\bullet \Rightarrow (p,q) \notin F_P^+ \Rightarrow [p \notin P' \text{ or } q \notin P']$.

– a discrepancy. Analogously for $a^\bullet \cap P'$.

But this means that the pair (T',P') is an elementary net. From Corollary 7 and the obvious fact that MCH is a cover of P, we obtain that N is a proper net. ■

We shall say that two partial order relation $\leqslant_1$, $\leqslant_2$ are compatible iff:

1. $(\forall x,y)\ x\neq y \ \&\ (x,y)\in \leqslant_1 \Rightarrow (y,x)\notin \leqslant_2$,
2. $(\forall x,y)\ x\neq y \ \&\ (x,y)\in \leqslant_2 \Rightarrow (y,x)\notin \leqslant_1$.

Using the notion of compatibility of relations we can characterize nets of occurrences by means of elementary nets.

Lemma 10.

Let $N_1=(T_1,P_1)$, $N_2=(T_2,P_2)$, $N=(T_1\cup T_2,P_1\cup P_2)$ be s-nets, and let $(\forall p\in P_i)\ \mathrm{card}((^\bullet p)_{N_i})\leqslant 1$, $\mathrm{card}((p^\bullet)_{N_i})\leqslant 1$ for i=1,2.

Then $(\forall p\in P_1\cup P_2)\ \mathrm{card}((^\bullet p)_N)\leqslant 1 \ \&\ \mathrm{card}((p^\bullet)_N)\leqslant 1$.

Proof.

It is enough to notice that: $(\forall p\in P_i)\ (^\bullet p)_{N_i}=(^\bullet p)_N$ and $(p^\bullet)_{N_i}=(p^\bullet)_N$ for i=1,2. ■

Lemma 11.

Let N_1,N_2, $N=N_1\cup N_2$ be s-nets and let: $(F_{N_i})^+\cap \mathrm{id}=\emptyset$ for i=1,2.

Then $F_N^+\cap \mathrm{id}=\emptyset \iff F_{N_1}^*$, $F_{N_2}^*$ are compatible.

Proof.

$\Leftarrow$. $(F_{N_1})^+\cap \mathrm{id}=\emptyset \ \&\ (F_{N_2})^+\cap \mathrm{id}=\emptyset$ means that $F_{N_1}^*$, $F_{N_2}^*$ are partial order relations, but the union of two compatible partial order relations is also the partial order relation, then F_N^* is a partial order relation. But F_N^* is partial order relation iff $F_N^+\cap \mathrm{id}=\emptyset$.

$\Rightarrow$. Assume that $F^*_{N_1}$, $F^*_{N_2}$ are not compatible. Let $(x,y) \in F^*_{N_1}$, $(y,x) \in F^*_{N_2}$ and $x \neq y$. This means that $(x,x) \in F^+_N$, so $F^+_N \cap id \neq \emptyset$. ■

Theorem 12.

A simple net N is an occurrence net $\Leftrightarrow$ there is a set of elementary occurrence nets (chains of occurrences) $\{N_1,\ldots,N_m\}$ such that:

1. $N = N_1 \cup \ldots \cup N_m$,
2. $(\forall i,j=1,\ldots,m)$ $F^*_{N_i}$, $F^*_{N_j}$ are compatible.

Proof.

$\Rightarrow$ This follows from Theorem 9. It is enough to put $\{N_1,\ldots,N_m\} = elem(N)$, and to observe that $F^*_{N_i} = F^*_N|N_i$.

$\Leftarrow$ This is a consequence of Lammas 10 and 11. ■

Corollary 13 .

If $N=(T,P)$ is an occurrence net and $N'=(T',P') \sqsubseteq N$ then N' is also an occurrence net.

Proof.

Directly from the definition. ■

Corollary 14.

For every occurrence net N: $elem(N) = atoms(N)$.

Proof.

Of course $elem(N) \leq atoms(N)$. Let $N' \in atoms(N)$. Of course $N' \sqsubseteq N$, then by Corollary 13 we have that N' is an occurrence net. But it is easily seen that every atom which is the net of occurrences is elementary. ■

An application of the approach presented above can be found in [1,2]. Detailed properties of marked simple nets will be published soon.

Acknowledgements.

The author wish to thank A.Mazurkiewicz and E.Knuth for stimulating papers and discussions.

References.

[1] Janicki R., Synthesis of Concurrent Schemes, Lecture Notes in Comp. Sci. 64, Springer-Verlag, 1978, 298-307.

[2] Knuth E., Petri Nets and Trace Languages, Proc. of the 1st European Conf. on Parallel and Distributed Processing, Toulouse, 1979.

[3] Mazurkiewicz A., Concurrent Program Schemes nad Their Interpretations, DAIMI PB-78, Aarhus Univ. Publ., 1977.

[4] Petri C.A., Nichtsequentielle Prozesse, ISF Report 75-207, GMD Bonn, 1976.

ON THE ALGEBRAIC APPROACH TO DISTRIBUTED COMPUTER SYSTEMS.

Jan R. Just
Warsaw Technical University

1. Introduction.

In the paper an algebraic method of analysis and synthesis of distributed computer systems /DCS/ is considered. In paper Just [6] has been introduced, basic and general model for describing the /input-output/ behavior of such systems.

A distributed computer system can be considered from either the hardware viewpoint or the software viewpoint. From the hardware viewpoint it is a collection of processors connected over digital communication system. From the software viewpoint it is a distributed facility for the sharing of resources, facilities and information. Since the overall design goal of distributed computer systems is an integrated hardware and software system which satisfies certain performance requirements and design constrains, it is necessary to take into consideration both the hardware and software viewpoint.

In this paper we use "process" as the basic unit in our description of DCS, and as a mathematical model of the process we shall mean finite-control /FC-/ algorithm [8] , [9] . The model of FC-algorithm, introduced by Mazurkiewicz and later discussed by others, is a convenient mathematical tool for the investigation of properties of iterative processes.

Our description of system include coroutine machanism / Janicki, Just [5] /. By coroutines we mean program components able to interact in symetric fashion / Conway [2] /.
This concept is frequently applied in operating systems. /Brinch-Hansen [1] /.

Formally, our model is based on the notion of so called vector of coroutines. This notion was introduced by Janicki [3] , in order to describe the semantics of programs with coroutines.

The system presented in this paper might be mean as model for operating system working in "asynchronous parellel" where several processing units cooperate.

The main subject of this paper is the problem of processes synthesis in distributed computer systems. This problem is the solution of the following question.
We have a given multi-processors system and given process /program/, and we want to execute this program in this system. With multi-processors CS are connected some constrains as a result impossibility of the execution some a particular parts of process by given processor. On the other hand, a process executed in MCS has some requirements such as facilities of the system.
The problem of synthesis of processes in DCS will be such a distribution of processes in the system / such an allocation of tasks to particular processors /, that execution of the particular process will be feasible take into consideration these reguirements and constrains.

The subject of distributed processes synthesis gains more and more interest, which is, of course, connected with developments in microprocessor technology as a future implementation tool.

2. Basic notations.

In this section we adopt for our purposes some notions from the formal language theory, set theory and abstract algebra.

Let X be a set. By Rel(X) we denote the set of $Rel(X)=\{R \mid R \subseteq X \times X\}$ and by id we denote the identity relation in Rel(X) . For each relation R, let $R^0=id$. For $R \in Rel(X)$, $R^* = \bigcup_{i=0} R^i$, $R^+ = \bigcup_{i=1} R^i$. The relation R^* is called the reflexive transitive closure of R, and R^+ is called the transitive closure of R.

By $\mathcal{P}(X)$ we denote the power set of X, and by card(X) the cardinality of X. By $[n]$ we denote the set $\{1,2,\ldots,n\}$.

Let Σ be an alphabet. A word over an alphabet is a finite seguence of elements in Σ . The word of length zero is denoted by ε . The set of all words, including ε , over an alphabet Σ is denoted by Σ^* . We put also $\Sigma^+ = \Sigma^* - \{\varepsilon\}$, $\Sigma^\cdot = \Sigma \cup \{\varepsilon\}$.

A right linear grammar is a 4-tuple:

$G = (\Sigma, V, \sigma, P)$, where:

/i/ Σ is a finite set of terminal symbols,

/ii/ V is a finite set of nonterminal symbols,

/iii/ $\sigma \in V$ is the start symbol,

/iv/ P is a finite subset of $V \times \Sigma^* V^*$, i.e. P is a finite set of ordered pairs (a,Rb), where $a \in V$, $R \in \Sigma^*$, $b \in V^*$. Elements (a,Rb) in P are called productions and are written $a \to Rb$. For each y,z in $(\Sigma \cup V)^*$ write $y \Rightarrow z$ if there exists u_1, u_2, u and v such that $y = u_1 u u_2$, $z = u_1 v u_2$ and $u \to v \in P$.

Let $L(G) = \{ w \mid w \in \Sigma^* \wedge \sigma \overset{*}{\Rightarrow} w \}$. This language is called the language generated by the grammar G.

By a net we mean any system $N = (U, \leq, \circ, e, \underline{0})$, where:

/i/ $(U, \leq)$ is a complete lattice,

/ii/ $(U, \circ, e, \underline{0})$ is a monoid with unity e, and with zero $\underline{0}$,

/iii/ a binary relation $\circ$ is additive and continuous.

The notion of nets was introduced by Blikle. Basic examples of nets are the following, the net of languages and the net of binary relations. The net of languages is particulary interesting in our considerations.

By a net of languages over the alphabet Σ we mean the system $(\mathcal{P}(\Sigma^*), \subseteq, \circ, \{\varepsilon\}, \emptyset)$, where $\circ$ is a concatenation of languages. For more details on the subject of nets the reader is advised to refer to [9].

By a finite-control /abbr. FC-algorithm/ over net N we mean any system: $A = (V_A, a_1, \varepsilon, P_A)$, where:

/i/ $V_A = \{ a_1, a_2, \ldots, a_k \}$ is an arbitrary set of control symbols,

/ii/ $a_1 \in V_A$ is the start symbol,

/iii/ ε is the terminal /empty/ symbol,

/iv/ P_A is a finite set of triples of the form (a_i, r, a), where $a_i \in V_A$, $a \in V_A^*$, $r \in U$. More details and the interpretation of Mazurkiewicz algorithms can be found in [8], [9].

By a vector of coroutines over net N we mean any system

$C = (i_0, A_1, \ldots, A_n)$, where:

/1/ i_0 is an integer / $1 \leq i_0 \leq n$ / called the number of initial component,

/2/ A_i for $i = 1, \ldots, n$ are triples /called components/

$A_i = (V_i, \sigma_i, P_i)$, where:

/i/ V_i is an alphabet / of control symbols of A_i /,
/ii/ $\sigma_i \in V_i$ is the start symbol of A_i,
/iii/ P_i is a finite subset of the set:
$(\{i\} \times \{1,2,\ldots,n\}) \times (V \times V^*) \times U$.

For more details on the subject of vector of coroutines the reader is advised to refer to [3] , [4] .

3. The model of the distributed computer system.

By a model of DCS we shall mean 3-tuple:

DCS = (S, MP, AL) , where:

S - a structure of a system,
MP - a set of processes in a system,
AL - a mapping AL : MP→S.

3.1. Structure of DCS.

By a structure of DCS we mean a directed graph:

$S = (N, n_0, LT)$, where:

N - a set of nodes / stations of network /,
$n_0 \in N$ - an initial node,
$LT \subseteq N \times N$ - a set of transmission lines.

Example 3.1.1.

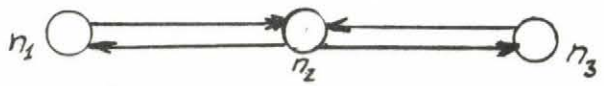


Fig. 1

$N=\{n_1,n_2,n_3\}$ $n_0=n_1$, $LT=\{(n_1,n_2);(n_2,n_1),(n_2,n_3),(n_3,n_2)\}$.

□

3.2. Processes in DCS.

A task realization in DCS is the result of activity of processes distributed in system, connected asynchronously. During task realization the user of system create so called the virtual network of processes. The virtual network of processes consists of a set of logically connected processes. Each of coprocess for a given virtual process is executed in different processor of DCS.

In order to describe the set of processes in DOS we shall introduce a mathematical object, called a matrix of coprocesses.
This object describe the algorithmic structure /semantics/ of DOS.

3.2.1. Matrix of coprocesses.

By a matrix of coprocesses over the net N we mean a system:

$$MP = (\mathcal{A}, I_0) \quad , \text{ where:}$$

$$\mathcal{A} = \{A_{ij}\}_{\substack{i\in[m]\\ j\in[n]}} \quad , \qquad I_0 \in [m]\times[n] \quad .$$

A_{ij} - are coprocesses / see below /, I_0 - indicate the start process.

The set $\mathcal{A}$ can be interpreted as a matrix :

$$\mathcal{A} = \begin{bmatrix} A_{11}, \dots, A_{1n} \\ \dots\dots\dots \\ A_{m1}, \dots, A_{mn} \end{bmatrix} \begin{matrix} \leftarrow \\ \\ \leftarrow \end{matrix} \begin{bmatrix} \text{process 1} \\ \dots \\ \text{process m} \end{bmatrix}$$

Each line in the above matrix represent one process.

A_{ij} - a triple which represent j-th coprocess in i-th process.

$$A_{ij} = (V_{ij}, \sigma_{ij}, P_{ij}) \quad \text{or} \quad A_{ij} = (\emptyset, \{\varepsilon\}, \emptyset)$$

where:

/i/ V_{ij} - an alphabet /of control symbols of A_{ij}/,
/ii/ $\sigma_{ij} \in V_{ij}$ - the start symbol of A_{ij},
/iii/ P_{ij} - a finite subset of the set:

$$(i,j)\times([m]\times[n])\times(V\times V^*)\times U.$$

This mean that P_{ij} is a finite set of 4-tuples of tha form:

$$(i\to r, j\to s, a\to b, R) \quad , \text{ where:}$$

1/ $i\to r \in \{i\}\times[m]$,
2/ $j\to s \in \{j\}\times[n]$,
3/ $a\to b \in V\times V^*$,
4/ $R \in U$.

P_{ij} is called the set of instructions of A_{ij}. Let $P = \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} P_{ij}$.
Each instruction consists of four parts:

1/ $i\to r$ indicates the process which will be active after the execution of the instruction / r-th process will be active/,

2/ $j\to s$ indicates the coprocess which will be active after tne execution of the instruction,

3/ $a\to b$ indicates the current /a/ and next /b/ point of A_{ij} component,

4/ R the "action" of the instruction.

R- in respect to an abstract character, we shall mean as the program, the part of the program or an activity of the operating system.

Let $E_{ij} = \{R \in U \mid \exists (i \to r, j \to s, a \to b, R) \in P_{ij}\}$ for $i \in [m], j \in [n]$.
The set of actions of the matrix of coprocesses we denote by E_M.

$$E_M = \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} E_{ij}.$$

Let $ms = \bigtimes_{i=1}^{m} \bigtimes_{j=1}^{n} V_{ij}$. / $\bigtimes$ - a cartesian product /.

Each an element $\alpha \in ms$ we shall describe in the following form:

$$\alpha = \begin{bmatrix} a_{11}, \dots, a_{1n} \\ \dots\dots\dots \\ a_{m1}, \dots, a_{mn} \end{bmatrix}, \text{ where: } (\forall i \leq m, j \leq n)\ a_{ij} \in V_{ij}^{\bullet}.$$

The set $MS = [m] \times [n] \times ms$ is called the set of control states of the matrix MP.

Let: $co: [m] \times [n] \times ms \to \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} V_{ij}^{\bullet}$ be a function such that:
$co(i,j,\alpha) = a_{ij}$.

Each $(i \to r, j \to s, a \to b, R)$ can be regarded as a relation in the set $Rel(MS)$ defined in the following way: $y_1 (i \to r, j \to s, a \to b, R)\, y_2 \Leftrightarrow$

$\Leftrightarrow (\exists \alpha, \beta \in ms)\ y_1 = (i,j,\alpha)$, $y_2 = (r,s,\beta)$ and $co(i,j,\alpha) = a$, $co(r,s,\beta) = b$.

The set $MT = \{(i,j,\alpha) \in MS \mid co(i,j,\alpha) = \varepsilon\}$ is called the set of terminal control states of MP.

We put $y_0 = (i_0, j_0, \alpha_0)$, where:

$$co(i,j,\alpha_0) = \begin{cases} \sigma_{ij} & \text{for } A_{ij} \neq \Theta \\ \varepsilon & \text{for } A_{ij} = \Theta \end{cases}$$

/ By Θ is denoted of empty coprocess of the form $(\emptyset, \{\varepsilon\}, \emptyset)$ /
The control state y_0 is called the start control state of MP.

Consider now a finite sequence of elements P_M / instructions/, such that there exists a sequence of control states: $y_1, \dots, y_{m+1}$ /elements of MS /, with the following properties:

1/ $\forall (k \leq d)\ y_k (i_k \to r_k, j_k \to s_k, a_k \to b_k, R_k)\, y_{k+1}$,

2/ $y_{d+1} \in MT$.

Each such a sequence of instructions will be called an y_1-computation.
To above sequence of control states corresponds the sequence of actions: $(R_1, \dots, R_d)$.
The corresponding sequence of actions /i.e. elements of E_M/ is called an y_1-trace.
The finite set of all sequences of control states we denote by $Tr_M(y)$,
Let $Tr(M) = \bigcup \{Tr_M(y) \mid y \in MS\}$,
and function $\mathcal{H}$: $Tr(M) \to U$ let be defined by:

$$\mathcal{R}(\langle R_1, \ldots, R_d \rangle) = R_1 \circ R_2 \circ \ldots \circ R_d$$

for $\forall (R_1, \ldots, R_d) \in Tr(M)$.

Each element from Tr(y)/trace/ produce its own outcome therefore the outcome of the whole set will be the join

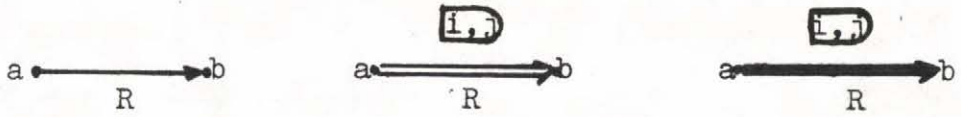
$$Tail_M(y) = \bigcup \{ \mathcal{R}(t) \mid t \in Tr(M) \} .$$

The concept of Tail element was introduced originally by Mazurkiewicz for sequential processes [8] .

In our model the vector $Tail_M(y_0)$ expressed the outcome of the virtual process in DCS.

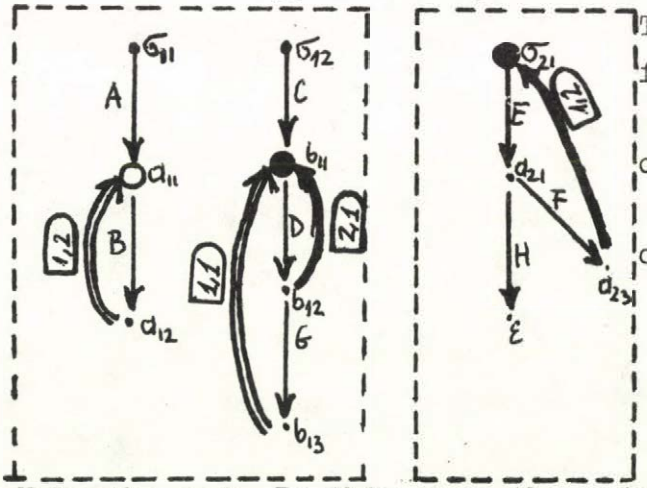
The matrix of coprocesses is useful where the number of processes and coprocesses is finite and known, and processes are iterative.

Every matrix of coprocesses can be represented graphicaly using graphs:



to denote instructions $(i \to i, j \to j, a \to b, R)$, $(i \to i, j \to s, a \to b, R)$ and $(i \to r, j \to s, a \to b, R)$ respectively.

Example 3.2.1.1. Consider the system of processes which consists of two processes such that the first process consists of two coprocesses. Let this syytem be represented by the below flowdiagram.



The interpretation of the above flowdiagram is the following.

o - the reactivation point of coprocess.

● - the reactivation point of the process.

Let N be the net. In this case the matrix of coprocesses can be defined as follows:

$$MP = (\mathcal{A}, I_0) \quad , \text{ where:}$$

$$\mathcal{A} = \{ A_{ij} \}_{\substack{i \in [1,2] \\ j \in [1,2]}} \quad , \quad I_0 = (1,1)$$

where:

$$A_{11}=(\{\sigma_{11},a_{11},a_{12}\},\sigma_{11},P_{11})$$

$$P_{11}=\{(1\to1,1\to1,\sigma_{11}\to a_{11},A),(1\to1,1\to1,a_{11}\to a_{12},B),$$
$$(1\to1,1\to2,a_{12}\to a_{11},e)\}$$

$$A_{12}=(\{\sigma_{12},b_{11},b_{12},b_{13}\},\sigma_{12},P_{12})$$

$$P_{12}=\{(1\to1,2\to2,\sigma_{12}\to b_{11},C),(1\to1,2\to2,b_{11}\to b_{12},D),$$
$$(1\to2,2\to2,b_{12}\to\sigma_{12},e),(1\to1,2\to2,b_{12}\to b_{13},G),$$
$$(1\to1,2\to1,b_{13}\to b_{11},e)\}$$

$$A_{21}=(\{\sigma_{21},a_{21},a_{23}\},\sigma_{21},P_{21})$$

$$P_{21}=\{(2\to2,1\to1,\sigma_{21}\to a_{21},E),(2\to2,1\to1,a_{21}\to\varepsilon,H),$$
$$(2\to2,1\to1,a_{21}\to a_{23},F),(2\to1,1\to1,a_{23}\to\sigma_{21},e)\}$$

$$A_{22}=\Theta$$

In this example:
the set of an actions is following: $E_M=E_{11}\cup E_{12}\cup E_{21}=\{A,B,C,D,E,F,G,H,e\}$
the set of control states $MS=\{1,2\}\times\{1,2\}\times ms$, where:

$$ms=\begin{bmatrix}\{\sigma_{11},a_{11},a_{12}\} & \{\sigma_{12},b_{11},b_{12},b_{13}\}\\ \{\sigma_{21},a_{21},a_{23}\} & \emptyset\end{bmatrix}$$

the start control state of MP: $y_0=(1,1,\alpha_0)$,where:

$$\alpha_0=\begin{bmatrix}\sigma_{11} & \sigma_{12}\\ \sigma_{21} & \varepsilon\end{bmatrix}$$

□

3.2.2. Semantics of the matrix of coprocesses.

Proving properties of the system of processes / in our model / is proving properties of the vector $Tail_M$. Thus this vector must be found explicitly. It is possible by proving that $Tail_M$ is the least solution of so-called canonical set of equations for each given matrix of coprocesses.
Analysis of the vector Tail and the method of solving the canonical set of equations are adopted from [3], [4], [8], [9].

Let $MP = (\mathcal{A}, I_0)$ be an arbitrary matrix of coprocesses over the net N.
Let:

/1/ $VR_{ij} = \{b \in V'_{ij} \mid \exists (i \to r, j \to s, a \to b, R) \in P_{ij}\ \mathcal{R}(i,j) \neq (r,s)\} \cup \{\sigma_{ij}\}$
if $A_{ij} \neq \Theta$ and $(i,j) \neq (i_0, j_0)$

/2/ $VR_{ij} = \{b \in V'_{ij} \mid \exists (i \to r, j \to s, a \to b, R) \in P_{ij}\ \mathcal{R}(i,j) \neq (r,s)\}$
if $A_{ij} \neq \Theta$ and $(i,j) \neq (i_0, j_0)$

/3/ $VR_{ij} = \{\varepsilon\}$
if $A_{ij} = \Theta$.

Elements of the set VR_{ij} / for $i=1,\ldots,m$, $j=1,\ldots,n$ / will be called <u>reactivation points</u> of the component A_{ij}.
Reactivation points of the matrix of coprocesses component correspond to reactivation points of the coprocesses system. They simply are these control symbols /labels/ from which components are resumed.

Let, for each $i \in [m]$, $j \in [n]$:

$$ms_{ij} = \bigcup_{r=1}^{m} \bigcup_{s=1}^{n} W_{rs}$$

where:

$$W_{rs} = \begin{cases} VR_{rs} & r \neq i \text{ or } s \neq j \\ V'_{rs} & r = i \text{ and } s = j \end{cases}$$

Let $MSP = [m] \times [n] \times \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} ms_{ij}$.

Elements of MSP will be called <u>proper control states</u> of MP.

For every set X by seq(X) we denote the set of all sequences of elements of X.

Let $t \in seq(X)$. We shall write $a \sqsubset t$, if $t = (t_1, a, t_2)$, where $a \in X$ and $t_1, t_2 \in seq(X)$.

<u>Lemma 3.2.2.1.</u>
If $z \in MSP$ then $\{y \mid y \sqsubset z\text{-statetrace}\} \subseteq MSP$. □

Note that $y_0 = (i_0, j_0, \alpha_0) \in MSP$.
From this and lemma3.2.2.1, it follows that all elements of all y_0-statetrace belong to MSP.
Thus, in order to describe $Tail_M(y_0)$ we can restrict our considerations to the set MSP. This set is finite by the definition.
Assume that card (MSP) =d.

Now we formulate the fixpoint semantics of the matrix ML.
Let:

$$\mathcal{F} : MSP \times MSP \to U$$

$$\mathcal{G} : MSP \to U$$

be functions, defined in the following way:

if $y_1 = (i, j\alpha)$, $y_2 = (r, s, \beta)$, then:

$$\mathcal{F}(y_1, y_2) = \begin{cases} \bigcup \{ R \mid (i \to r, j \to s, co(i,j,\alpha) \to co(i,j,\beta) , R) \in P_{ij} \\ \quad \text{if } (\forall t \neq i, k \neq j)\ co(t,k,\alpha) = co(t,k,\beta) \} \\ \underline{0} \quad \text{elsewhere} \end{cases}$$

and if $y = (i,j,\alpha)$, then:

$$\mathcal{G}(y) = \begin{cases} e \text{ if } co(i,j,\alpha) = \mathcal{E} \\ \underline{0} \text{ if } co(i,j,\alpha) \neq \mathcal{E} \end{cases} .$$

We can put: $MSP = \{ z_1, \ldots, z_d \}$, where: $z_1 = y_0$. Note that we can number coordinates of U^d with elements of MSP, and instead of $(X_1, \ldots, X_d)$ we can write $(X_{z_1}, \ldots, X_{z_d})$.

The canonical set of equations of matrix of coprocesses is following:

$$X_{z_1} = \bigcup_{z \in MSP} \mathcal{F}(z_1, z) \cdot X_z \cup \mathcal{G}(z_1)$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$X_{z_d} = \bigcup_{z \in MSP} \mathcal{F}(z_d, z) \cdot X_z \cup \mathcal{G}(z_d)$$

Writing this set down for any particular matrix of coprocesses we clearly omit components with $\mathcal{F}(z_k, z) = \underline{0}$ and $\mathcal{G}(z) = \underline{0}$.

Theorem 3.2.2.1. For every matrix of coprocesses MP the vector

$$(Tail_M(z_1) , \ldots , Tail_M(z_d))$$

is the least solution of the canonical set of equations of MP. □

They equations are usually solved by means of the variable elemination method. The particular description of that method can be found in [9] . The present approach extends the theory from [9] , [3] to matrices of coprocesses.

A given set of equations / in that paper they are omited / can be transformed to an equivalent flowchart.

Example 3.2.2.1./ continuation of an example 3.2.1.1 /

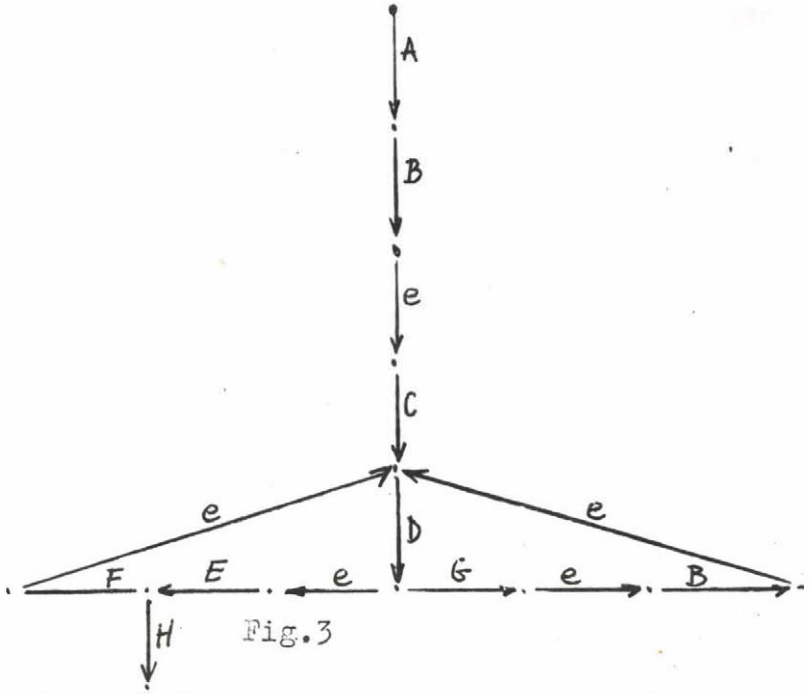


Fig.3

The solution of the canonical set of equations of given matrix MP is:

$Tail_M(y_0) = ABCD((EF \cup GB)D)^* EH.$

□

The graph from Fig.3 creates so called the virtual process graph G.

3.3. Mapping AL.

The mapping AL is the third element of the DCS model. To describe a particular system, it is necessary to specify:

1/ how to allocate processes to processors,

2/ how to allocate communication lines between that processors.

It is specified by mapping AL.

The structure of DCS we was defined as graph: $S = (N, n_0, LT)$ / see 3.1 /. Processes in DCS creates the virtual process graph G. This graph can be expressed in following way: $G = (MS, y_0, P)$, where $P \subseteq MS \times MS$.

The mapping AL is a some homomorphism between structure of DCS S and the graph of given virtual process G.

It s mean, that the structure of logical channels between the components of the given process, must be adequate to the structure of connections between processors of DCS.

Specifing the mapping AL we must take into consideration the "technical conditions" of DCS, and requirements of the executed process components.

4. Synthesis of distributed computer systems.

The problem of synthesis of distributed computer systems is the solution of the followig question.
We have a given multi-processor system and given process , and we want to execute this process /program/ in this system. With multi-processor CS are connected some constrains as a result impossibility of the execution some particular parts of the process by a given processor. On the other hand, process executed in DCS has some requirements such as facilities of the system.
The problem of synthesis of processes in DCS will be such a distribution of processes in the system / such an allocation of tasks to particular processors /, that execution of the particular process will be feasible take into consideration these requirements and constrains.

To solve this problem in formal way, we shall now present some properties of regular expressions and regular languages. It is necessary to further considerations.

4.1. Properties of regular expressions and regular languages.

In this section we recall some theory from Janicki [4].
The set of regular expressions over an alphabet Σ, $REX(\Sigma)$, we shall mean the least set of terms which fulfils the following conditions:

1. $\Sigma^* \subseteq REX(\Sigma)$,
2. $(\forall u \in REX(\Sigma))$ $u^*, (u) \in REX(\Sigma)$,
3. $(\forall u_1, u_2 \in REX(\Sigma))$ $u_1 u_2$, $u_1 \cup u_2 \in REX(\Sigma)$.

For every regular expression WR, let $|WR|$ denote the regular language defined by WR.

A family of sets $\{\Sigma_1, \Sigma_2, \ldots, \Sigma_n, \Sigma\}$ such that $\Sigma_1 \cup \ldots \Sigma_n = \Sigma$, and $(\forall\ i,j=1,2,\ldots,n)\ i \neq j \Rightarrow \Sigma_i \cap \Sigma_j = \emptyset$ will be called the n-partition of the alphabet .

Note that every regular expression over Σ is a word from $(\Sigma \cup \{(,), \cup, *\})^*$. By a segment of the word $w = w_1 w_2 \ldots w_m$ we shall mean every word of the form: $w_k w_{k+1} \ldots w_{k+p}$, where $1 \leq k \leq n, k+p \leq m$.

By a factor of the regular expression WR we mean any maximal segment of WR which does not contain characters (,) , ∨ and does not begin from the character * . Every factor can be treated as a word over the alphabet $\Sigma \cup \{*\}$. Let fac(WR) denote the set of all factors of the regular expression WR. For example, if WR=ABCD((EF∨GB)D)*EH then fac(WR) = {ABCD, D, EF, GB, EH} .

For every alphabet Σ, let Σ^* be the alphabet defined as follows: $\Sigma^* = \Sigma \cup \{A^* \mid A \in \Sigma\}$, where every two characters A, * written of the form A^* are treated as one symbol.

Let $u \in (\Sigma^*)^+$. The word u can be unambiguously represented as the concatenation: $u=u_1 \ldots u_m$, where:

1. $(\forall k \leq m)\ (\exists\, i_k \leq n) \qquad u_k \in (\Sigma^*_{i_k})^+$,

2. $(\forall\, 1 \leq k \leq m) \qquad u_{k-1}u_k \notin (\Sigma^*_{i_k})^+$.

Example 4.1.1. Let $u = ABCD^*G^*B^*EH \in (\Sigma^*)^+$, where: $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 = \{A,B\} \cup \{C,D,G\} \cup \{E,H\}$.
In this case: $u=u_1u_2u_3u_4$, where: $u_1=AB \in (\Sigma^*_1)^+$, $u_2=CD^*G^* \in (\Sigma^*_2)^+$, $u_3=B^* \in (\Sigma^*_1)^+$, $u_4=EH \in (\Sigma^*_3)^+$. □

If $u_1 \ldots u_m$ is a decomposition of the word u of the form described above, and for i=1,...,n : $\Lambda_i = \{\lambda_{i1}, \ldots, \lambda_{ii-1}, \lambda_{ii+1}, \ldots, \lambda_{in}\}$, $\Lambda = \bigcup_{i=1}^{n} \Lambda_i$, then let $\lambda(u) \in (\Sigma \cup \Lambda)^*$ be the word defined as follows: $\lambda(u) = u_1 \lambda_{i_1 i_2} u_2 \lambda_{i_2 i_3} \ldots \lambda_{i_{m-1} i_m} u_m$.

Example 4.1.2. If u is the word defined in Example 4.1.1., then

$$\lambda(u) = AB\,\lambda_{12}CD^*G^*\lambda_{21}B^*\lambda_{13}EH.$$

□

Let $T_1 = \{f_1, f_2, \ldots, f_n\}$ be the family of mappings from $(\Sigma^*)^+$ into $(\Sigma^* \cup \Lambda)^*$ defined in the following way:

$$(\forall i \in [n]) \quad f_i(u) = \begin{cases} \lambda(u) & i=i_1=i_m \\ \lambda_{ii_1}\lambda(u) & i_1 \neq i \ \&\ i_m=i \\ \lambda(u)\lambda_{i_m i} & i_1=i \ \&\ i_m \neq i \\ \lambda_{ii_1}\lambda(u)\lambda_{i_m i} & i_1 \neq i \ \&\ i_m \neq i \end{cases}$$

Example 4.1.3. / compare example 4.1.2. / For $f_1 \in T_1 = \{f_1, f_2, f_3\}$

$$f_1(u) = AB\,\lambda_{12}CD^*G^*\,\lambda_{21}B^*\lambda_{12}EH\,\lambda_{31}.$$

□

Note that every regular expression $WR \in REX(\Sigma)$ is the word over the alphabet $\Sigma \cup \{(,), \cup, *\}$ and it can be defined as the concatenation:

$$WR = v_0 u_1 \ldots u_m v_m \text{, where:}$$

$$v_0 \in \{(\}^*,$$

$$(\forall\, i \in [m]) \quad u_i \in fac(WR),$$

$$(\forall\, i \in [m-1]) \quad v_i \in \{(,), \cup, *\}^*,$$

$$v_m \in \{), *\}^*.$$

Let $T_2 = \{F_1, F_2, \ldots, F_n\}$ be the family of mappings described on $REX(\Sigma)$ in the following way:

$$(\forall\, WR = v_0 u_1 v_1 \ldots u_m v_m \in REX(\Sigma),\ \forall\, i \in [n])$$

$$F_i(WR) = v_0 f_i(u_1) v_1 \ldots f_i(u_m) v_m.$$

Corollary 4.1.1. $(\forall\, i \in [n])\ h_\lambda(|F_i(WR)|) = |WR|$.

□

Example 4.1.4. It can be formally prove that for the regular expression $WR = ABCD((EF \cup GB)D)^*EH$ and for the partition of alphabet

$$\Sigma = \{A, B\} \cup \{C, D, G\} \cup \{E, H, F\} = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 :$$

$$WR_1 = F_1(WR) = AB\,\lambda_{12}CD\,\lambda_{21}((\lambda_{13}EF\,\lambda_{31} \cup \lambda_{12}G\,\lambda_{21}B)\lambda_{12}D\,\lambda_{21})^*\lambda_{13}EH\,\lambda_{31}.$$

□

Note that with any regular expression WR we can associate the regular expression WR' which has the following properties:

1. $|WR'| = |WR|$, and

2. $$WR' = \begin{cases} Q_1P_1 \cup Q_2P_2 \cup \ldots \cup Q_mP_m \cup \varepsilon & \varepsilon \in |WR|, \\ Q_1P_1 \cup Q_2P_2 \cup \ldots \cup Q_mP_m & \varepsilon \notin |WR| \end{cases}$$

where $P_i \in fac(WR)$, $Q_i \in REX(\Sigma)$ for $i = 1, \ldots, m$.

/ It s follows from the fact $\alpha^* = \alpha^*\alpha \cup \varepsilon$ ./

Define $NREX(\Sigma) = \{WR' \mid WR \in REX(\Sigma)\}$.

Let $T_3 = \{g_1, g_2, \ldots, g_n\}$ be the family of mappings described on by the following equality:

$$(\forall i\in[n])\ g_i(u)=\begin{cases}\lambda(u) & \text{if } i_1=i\\ \lambda_{ii_1}\lambda(u) & \text{if } i_1\neq i\end{cases}$$

Let $T_4=\{G_{11},G_{12},\ldots,G_{nn}\}$ be the family of mappings described on the regular expressions of form QP, where $Q\in REX(\Sigma)$, $P\in fac(QP)$.

$$(\forall i,j\in[n])\ G_{ij}(QP)=\begin{cases}F_i(Q)\ g_i(P) & P\in(\Sigma_j^*)^*\\ F_i(Q)\ g_i(P)\lambda_{kj} & P\in(\Sigma_k^*)^*\ \text{ł}\ k\neq j.\end{cases}$$

Example 4.1.5. Let $WR=ABCD((EF\vee GB)D)^*EH$ where: $\Sigma_1=\{A,B\}$,

$\Sigma_2=\{C,D,G\}$, $\Sigma_3=\{E,H,F\}$.

In this case: $P\in(\Sigma_3^*)^+$ and:

$$G_{11}(WR)=\underbrace{AB\lambda_{12}CD\lambda_{21}((\lambda_{13}EF\lambda_{31}\vee\lambda_{12}G\lambda_{21}B)\lambda_{12}D\lambda_{21})^*}_{F_1(Q)}\underbrace{\lambda_{13}EH\lambda_{31}}_{g_1(P)}$$

$G_{12}(WR)=\ldots\ldots$

$G_{13}(WR)=AB\lambda_{12}CD\lambda_{21}((\lambda_{13}EF\lambda_{31}\vee\lambda_{12}G\lambda_{21}B)\lambda_{12}D\lambda_{21})^*\lambda_{13}EH$

$\vdots$ □

Let $T_5=\{H_{11},H_{12},\ldots,H_{nn}\}$ be the family of mappings described on $NREX(\Sigma)$ by the following equality: / $\forall WR\in NREX(\Sigma)$/ , / $\forall i,j\in[n]$/

$$H_{ij}(WR)=\begin{cases}G_{ij}(Q_1P_1)\vee\ \ldots\vee G_{ij}(Q_mP_m) & WR=Q_1P_1\vee\ \ldots\vee Q_mP_m\\ G_{ij}(Q_1P_1)\vee\ \ldots\vee G_{ij}(Q_mP_m)\vee\varepsilon & WR=\ \text{—}\,\text{“}\,\text{—}\ \ \vee\varepsilon\ \text{ł}\ i=j\\ G_{ij}(Q_1P_1)\vee\ \ldots\ G_{ij}(Q_mP_m)\vee\lambda_{ij} & WR=\ \text{—}\,\text{“}\,\text{—}\ \ \text{ł}\ i\neq j\end{cases}$$

4.2. Synthesis of matrix of coprocesses.

Synthesis of the matrix of coprocesses is the solution of the following question.

The process given is in the form of the regular language / or the regular expression/ L or WR, over an alphabet Σ / the set of action symbols /.

Taking into consideration technical constrains as a resul impossibility of execution some particular parts of the process by a given processor we define the mapping AL.

This mapping create the partition of Σ $-\{\Sigma_{11}, \Sigma_{12}, \ldots, \Sigma_{m.n}, \Sigma\}$

for $\Sigma_{11} \cup \Sigma_{12} \cup \ldots \cup \Sigma_{m.n} = \Sigma$ and

$\forall (i,j)\ ,(r,s) \in [m] \times [n]\ \ (i,j) \neq (r,s) \Rightarrow \Sigma_{ij} \cap \Sigma_{rs} = \emptyset$.

For given the problem we should build a matrix of coprocesses which consists of $m \cdot n$ components and generates the language $|R|$ and such that the alphabet / of action symbols / of i,j -th component is contain in Σ_{ij}.

One should build a matrix of coprocesses $M = (\mathcal{A}, i_0)$ over the alphabet Σ with the following properties:

1. $(\forall (i,j) \in [m] \times [n])\ \ \Sigma_{ij} \subseteq \Sigma_{ij} \cup \{\varepsilon\}$
2. $\mathrm{Tail}_M(y_0) = L = L(MP)$

/ where $L(MP)$ is the language generated by the matrix of coprocesses- see below /.

4.2.1. Some extensions of the matrix of coprocesses.

The first part of this section concerns the construction of the language $L(MP) = \mathrm{Tail}_M(y_0)$ i.e. the language generated by the matrix of coprocesses.

Let $N = (\mathcal{P}(\Sigma^*), \subseteq, \circ, \{\varepsilon\}, \emptyset)$ be the net of languages over the alphabet Σ .

The set $\tilde{S} = MS \times \Sigma^*$ is called the set of states of MP / where MS is the set of control states of MP - see 3.2.1 /.

Let $T \subseteq \tilde{S} \times \tilde{S}$ be the relation defined by the equivalence:

$$(y_1,u_1)\ T\ (y_2,u_2) \Leftrightarrow$$
$$\Leftrightarrow [(\exists (i \to r, j \to s, a \to b, R) \in P)\ \ (y_1,y_2) \in MS\ \ \&\ \ u_2 = u_1 R]$$

Lemma 4.2.1.1. $L(MP) = \{ w \in \Sigma^* \mid (\exists y \in MT)\ (y_0, \varepsilon)\ T^*\ (y,w) \}$.

Proof:

Let $w \in \{ w \in \Sigma^* \mid (\exists y \in MT)\ (y_0, \varepsilon)\ T^*(y,w) \}$.

This mean that there exists a sequence of elements of $\tilde{S}$:

(y_1,w_1) , ... , (y_m,w_m) , where $y_1 = y_0$, $w_1 = \varepsilon$, $y_m \in MT$, $w_m = w$ and there exists a sequence of instructions:

$(i_1 \to r_1, j_1 \to s_1, a_1 \to b_1, R)\ \ldots\ (i_{m-1} \to r_{m-1}, j_{m-1} \to s_{m-1}, a_{m-1} \to b_{m-1}, R_{m-1})$,

such that , for $k = 1, \ldots, m-1$:

$y_k\ (i_k \to r_k, j_k \to s_k, a_k \to b_k, R_k)\ y_{k+1}$, and $w_{k+1} = w_k R_k$;

Note that $(R_1,\ldots,R_{m-1})$ is the y_1-trace and $w=R_1\ldots R_{m-1}$.
Thus, $w \in Tail_{M'}(y_1) = Tail_M(y_0) = L(MP)$.
Suppose that $w \in L(MP)$. Then there is y_0-trace of the form $(R_1,\ldots,R_m)$ such that $w=R_1\ldots R_m$.
Let $(y_1,\ldots,y_{m+1})$ be the a propriate y_0-statetrace, and let
$(i_1 \to r_1, j_1 \to s_1, a_1 \to b_1, R_1) \ldots (i_m \to r_m, j_m \to s_m, a_m \to b_m, R_m)$
be the appropriate y_0-computation.

Iut $w_1=\varepsilon$, $w_2=R_1,\ldots, w_m=R_1\ldots R_{m-1}$, $w_{m+1}=R_1\ldots R_m$.
Consider the sequence of pairs: (y_1,w_1) , ... , (y_{m+1},w_{m+1}) .
Obviously, for $k=1,\ldots,m+1$, $(y_k,w_k) \in \widetilde{S}$.
Notice, that by the definition of T, $(y_k,w_k)\ T\ (y_{k+1},w_{k+1})$ for $k=1,\ldots,m$, therefore:

$$(y_1,w_1)\ T^m (y_{k+1},w_{k+1}) \ .$$

Since $w_1=\varepsilon$, $w_{k+1}=w$, and $y_1=y_0$, then:

$$(y_0,\varepsilon) T^* (y_{k+1},w) \ .$$

From the fact that $(y_1,\ldots,y_{m+1})$ is the y_0-statetrace, we obtain $y_{m+1} \in MT$.
It turns out that $w \in \{w \in \Sigma^* \mid (\exists y \in MT)(y_0,\varepsilon)\ T^*(y,w)\}$.

□

The language L(MP) is called the language generated by the matrix of coprocesses MP. This language is interpreted as a description of the semantics of the matrix MR.

Proving properties of the system of processes / in our model / is proving properties of the language $L(MP) = Tail_M(y_0)$.
The language L(MP) does not contain much information about the structure of the matrix of coprocesses. If we know this language only we do not know anything about the number and the form of components.
Now we define a language which defines the language L(MP) , the number of components, and sublanguages defined by components.
Note that every component can be interpreted as certain context-free grammar.
Let $MP=(\mathcal{A}, I_0)$, where: $\mathcal{A}=\{A_{ij}\}_{\substack{i\in[m]\\ j\in[n]}}$, $I_0 \in [m]\times[n]$ and

$A_{ij}=(\Sigma_{ij}, V_{ij}, \sigma_{ij}, P_{ij})$ for $i=1,\ldots,m$, $j=1,\ldots,n$, be a matrix of coprocesses.

We define the following alphabets: $\Lambda_k = \{ \lambda_{kl_1}, \ldots, \lambda_{kl_m} \}_{\cdot n} - \{ \lambda_{kk} \}$,

$\Lambda'_k = \Lambda_k \cup \{ \lambda_k \}$, for:

$i,r \in [m]$, $j,s \in [n]$ and $k=(i,j)$, $l=(r,s)$.

Let also: $\Lambda = \bigcup_{k \in [m]\times[n]} \Lambda_k$, $\Lambda' = \bigcup_{k \in [m]\times[n]} \Lambda'_k$.

Let $\lambda(MP)$ be the matrix of coprocesses defined as follows:

$\lambda(MP) = (\mathcal{A}^\lambda, I_0)$, where: $\mathcal{A}^\lambda = \{ A^\lambda_{ij} \}_{\substack{i \in [m] \\ j \in [n]}}$, $I_0 \in [m]\times[n]$.

$A^\lambda_{ij} = (\Sigma_{ij} \cup \Lambda_{ij}, V_{ij} \cup \{\sigma'_{ij}\}, \sigma'_{ij}, P^\lambda_{ij})$ and

$P^\lambda_{ij} = \{ (i \to r, j \to s, a \to b, R\lambda_{kl}) \mid (i \to r, j \to s, a \to b, R) \in P_{ij} \ \& \ k \neq l \} \cup$

$\cup \{ (i \to r, j \to s, a \to b, R) \mid (i \to r, j \to s, a \to b, R) \in P_{ij} \ \& \ k = l \} \cup$

$\cup \{ (i \to i, j \to j, \sigma'_{ij} \to \sigma_{ij}, \mathcal{M})$ if $(i,j) = I_0$ then $\mathcal{M} = \lambda_{k_0}$, elsewhere $\mathcal{M} = \varepsilon$, whwre $k_0 = (i_0, j_0) \}$.

The language $L(\lambda(MP))$ contains all necessary information about the structure of the matrix MP.

Let $h_\Lambda : (\Sigma \cup \Lambda')^* \to \Sigma^*$ be the following homomorphism:

$$(\forall R \in \Sigma \cup \Lambda') \quad h_\Lambda(R) = \begin{cases} R & R \in \Sigma \\ \varepsilon & R \notin \Sigma \end{cases}$$

<u>Corollary 4.2.1.1.</u> $L(MP) = h_\Lambda (L(\lambda(MP)))$. □

<u>Proof:</u> Follows from definition of MP.

A component A_{ij} of MP is called <u>final</u> if there exists such an instruction $(i \to r, j \to s, a \to b, R) \in P_{ij}$ that $b = \varepsilon$.

The set of all final components of MP will be denoted by $FINAL_M$.

We restrict our attension to the matrix MP with the property $card\,(FINAL_M) = 1$.

For $i=1,\ldots,m$, $j=1,\ldots,n$, let $\lambda'(A_{ij})$ be a right linear grammar defined as follows:

$\lambda'(A_{ij}) = (\Sigma_{ij} \cup \Lambda_k, V_{ij}, \sigma_{ij}, \tilde{Q}_{ij})$, where: / $k=(i,j)$, $l=(r,s)$ /

$\tilde{Q}_{ij} = \{ a \to Rb \mid (i \to r, j \to s, a \to b, R) \in P_{ij} \} \cup \{ a \to R\lambda_{kl} b \mid (i \to r, j \to s, a \to b, R) \in P_{ij}$

$\& (i,j) \neq (r,s) \} \cup \{ a \to Rb\lambda_{kl} \mid (i \to r, j \to s, a \to b, R) \in P_{ij} \ \& \ A_{ij} \in FINAL_M \}$.

Let $L(\lambda'(A_{ij}))$ denote the language generated by the grammar $\lambda'(A_{ij})$. It can be prove, that if we know the grammars $\lambda'(A_{ij})$ / it's mean that we know the languages $L(\lambda'(A_{ij}))$ / we can define the language $L(\lambda(MP))$. Proof of this is long and arduous. Equivalent proof, for vector of coroutines, can be found in Janicki [4].

Now we fix I_0 - number of initial coprocess, / $I_0 = (i_0, j_0) \in [m] \times [n]$ /, and $I_F = (i_F, j_F) \in [m] \times [n]$ - number of final coprocess.

Let WR' be a regular expression from $NREX(\Sigma)$ such that $|WR'| = |WR|$. Define $\overline{WR} = H_{\hat{i}_0 \hat{j}_F}(WR')$, where H_{ij} is a mapping defined in the section 4.1, and $\hat{i} = num(c,e)$, $\hat{j} = num(d,g)$ / where : $(c \to d, e \to g, a \to b, R)$ is an instruction of MP / and $num: [m] \times [n] \to [n \cdot m]$ is defined in following way:
$num(i,j) = (i-1) \cdot m + j$ for $i \in [m]$, $j \in [n]$. Put $q = m \cdot n$.

As it was notified above, every regular expression is a word over the alphabet $\Sigma \cup \{(,), \vee, *, \varepsilon\}$. Thus, it can be an argument of the homomorphism h_Ω / which effaces characters belong to Ω /. From the formal language theory it follows that:
$(\forall \Omega \subseteq \Sigma)(\forall WR \in REX(\Sigma)) \; |h_\Omega(WR)| = h_\Omega(|WR|)$.
Let $\{\overline{WR}_{11}, \ldots, \overline{WR}_{mn}\}$ be the set of regular expressions of the following form: $(\forall num(i,j) \leq num(m,n)) \quad \overline{WR}_{ij} = h_{\Omega_{\hat{i}}}(\overline{WR})$, where:
$\Omega_{\hat{i}} = (\Sigma \cup \Lambda) - (\Sigma_{\hat{i}} \cup \Lambda_{\hat{i}})$.

Example 4.2.1.1./ see example 4.1.4 / Let $\hat{i}_0 = 1, \hat{i}_F = 3$. In this case: $WR' = WR$.
$\overline{WR} = AB\,\lambda_{12}CD\,\lambda_{21}((\lambda_{13}EF\,\lambda_{31} \vee \lambda_{12}G\,\lambda_{21}B)\,\lambda_{12}D\,\lambda_{21})^*\,\lambda_{13}EH$
$\Omega_1 = \{A, B, \lambda_{12}, \lambda_{13}\}$, $\Omega_2 = \{C, D, G, \lambda_{21}, \lambda_{23}\}$, $\Omega_3 = \{E, H, F, \lambda_{31}, \lambda_{32}\}$.
$\overline{WR}_{11} = AB\,\lambda_{12}((\lambda_{13} \vee \lambda_{12}B)\,\lambda_{12})^*\lambda_{13}$,
$\overline{WR}_{12} = CD\,\lambda_{21}((G\,\lambda_{21})D\,\lambda_{21})^*$,
$\overline{WR}_{21} = (EF\,\lambda_{31})^*\,EH$. □

From the formal language theory it also follows that for every $\hat{i} = 1, \ldots, q$ we can build a right linear grammar $G_{ij} = (\Sigma_{ij} \cup \Lambda_{ij}, V_{ij}, \sigma_{ij}, Q_{ij})$ such that:

1. $L(G_{ij}) = |\overline{WR}_{ij}|$,
2. $\forall a \to Rb \in \widetilde{Q}_{ij} \quad R \in \Sigma_{ij}^* \vee \Lambda_{ij}$.

Example 4.2.1.2. For expressions $\overline{WR}_{11}, \overline{WR}_{12}, \overline{WR}_{21}$ / see example 4.2.1.1. / these grammars can be described by the following graphs.

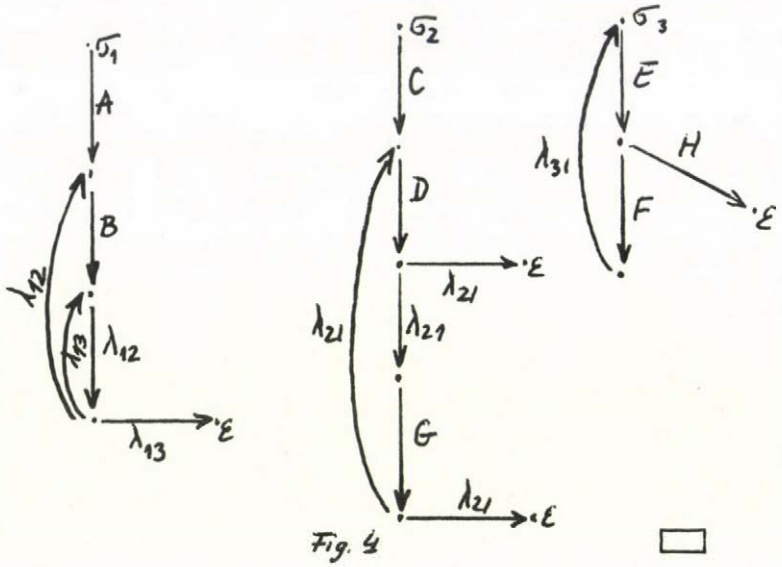


Fig. 4 □

Of course, the above construction is ambiguous, and for every $\hat{i}=1,\ldots,q$, if $\overline{WR}_{\hat{i}}$ contains the character * then there exist an infinite number of grammars which fulfil such conditions.

For a given regular expression WR, I_0, I_F and for given partition of the alphabet Σ let GRAM denote the family of all sets of grammars $\{G_{11},\ldots,G_{mn}\}$.

Let TAB be the following set of matrices of coprocesses.

$$MP \in TAB \Leftrightarrow (\exists \{G_{11},G_{12},\ldots,G_{mn}\} \in GRAM)$$

$G_{ij}=(\Sigma_{ij} \cup \Lambda_{ij}, V_{ij}, \sigma_{ij}, \tilde{Q}_{ij})$ and $MP=(\mathcal{A}, I_0)$, $\mathcal{A}=\{A_{ij}\}_{\substack{i\in[m]\\ j\in[n]}}$,

$I_0 \in [m]\times[n]$, $A_{ij}=(\Sigma_{ij} \cup \Lambda_{ij}, V_{ij}, \sigma_{ij}, P_{ij})$

$$P_{ij}=\{(i\to i, j\to j, a\to b, R) \mid a\to Rb \in \tilde{Q}_{ij} \wedge R \in \Sigma^{*}_{ij}\} \cup$$

$$\cup \{(i\to r, j\to s, a\to b, \varepsilon) \mid a\to \lambda_{num(i,j)\,num(r,s)}\, b \in \tilde{Q}_{ij} \wedge b \notin \Sigma\} \cup$$

$$\cup \{(i\to r, j\to s, a\to \varepsilon, \varepsilon) \mid a\to \lambda_{num(i,j)\,num(r,s)} \in \tilde{Q}_{ij} \wedge (i,j)=(i_F,j_F)\}.$$

Theorem 4.2.1.1. For every matrix of coprocesses $MP=(\mathcal{A}, I_0) \in TAB$

$\mathcal{A}=\{A_{ij}\}_{\substack{i\in[m]\\ j\in[n]}}$, $I_0 \in [m]\times[n]$:

1. $(\forall G_{ij} \in \{G_{11},\ldots,G_{mn}\} \in GRAM)\ G_{ij}=\lambda'(A_{ij})$
2. $L(MP) = |WR|$. □

Example 4.2.1.3. The matrix of coprocesses for our example /see example 4.2.1.2. /of system can be graphically expressed by the graphs from fig.5.

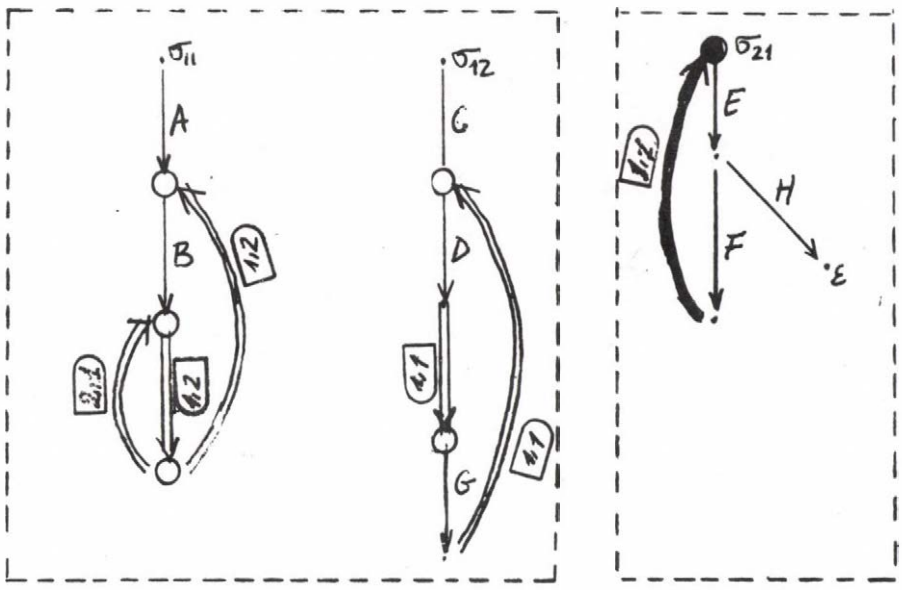


Fig. 5

4.2.2. The graph of the transmission of DCS.

Let's obtain so called the graph of transmission G_T, for a given matrix of coprocesses. The graph of transmission will represent the structure of logical connections between - allocated to separate processors - interacted coprocesses. Let $G_T=(N_T, L_T)$, where:
N_T - the subset of nodes of DCS structure - determined by mapping AL.
$L_T \subseteq N_T \times N_T$ - the set of logical channels between coprocesses of MP.
Having the expression $\overline{WR}$ /see 4.2.1. / we can obtain the set L_T.
/ The expression $\overline{WR}$ contain information about the language generated by the matrix of coprocesses and the information about resumptions of coprocesses./

Let $\mathcal{L}(x)$ be the function defined as follows:

$$(\forall x \in \Lambda^*)\mathcal{L}(x) = \{\lambda \in \Lambda \mid \exists u,v \in \Lambda^* \ , \ u\lambda v = x\} .$$

Now we define the set of symbols $\lambda_{ij} \in \Lambda$, belongs to expression $\overline{WR}$. This set will be denoted by $LAMB_{\overline{WR}}$.

$$LAMB_{\overline{WR}} = \mathcal{L}(h_\Sigma(\overline{WR})) .$$

We shall present an example illustrating the construction described above.

Example 4.2.2.1. Let $\overline{WR} = AB\lambda_{12}CD\lambda_{21}((\lambda_{13}EF\lambda_{31} \cup \lambda_{12}G\lambda_{21}B)\lambda_{12}D\lambda_{21})^*\lambda_{13}EH$.

In this case the set $LAMB_{\overline{WR}} = \{\lambda_{12}, \lambda_{21}, \lambda_{13}, \lambda_{31}\}$. □

Let $N_T = \{n_1, \ldots, n_{n \cdot m}\}$.

For a given matrixof coprocesses and a given allocation function AL :

$$(\forall\, i\in[m],\ j\in[n]) \qquad L_T=\left\{(n_i,n_j)\in N_T\times N_T \mid \exists\, \lambda_{ij}\in L\overline{MB}_{\overline{MR}}\right\}.$$

Example 4.2.2.2. The graph of transmission for the matrix of coprocesses from an example 4.2.1.3. has form:

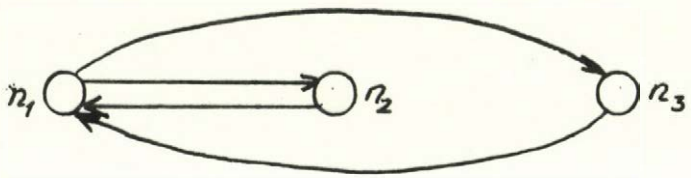


Fig.6 □

The connection structure of the graph G_T must be adequate to the connection structure of the grapg S - the structure of phisical connections between processors of the computer network. There have to exists the isomorphism between the graph G_T and the graph S. This requirements isn't satisfied in our an example system. / see example 3.1.1. and example 4.2.2.2 /
Thus, the matrix of coprocesses must be modified.

4.2.3. The modification of the matrix of coprocesses.

The definition of the graph of transmission of DCS is based on the expression $\overline{MR}$. Now this expression must be modified, such that the requirements of an isomorphism between graphs G_T and S will be satisfied. This modification will be discussed below.

Every the regular expression $\overline{MR}$ is an element of the set $REX(\Sigma^*, \Lambda')$. Let, for $1 \le i_q \le num(m,n)$, Λ'' be the alphabet:

$$\Lambda'' = \left\{\lambda_{i_1 i_2}, \lambda_{i_2 i_3}, \ldots, \lambda_{i_{q-1} i_q}\right\}$$

such , that:

$$(\forall\, \lambda_{a,b}\in\Lambda'')\ \exists((n_a,n_b)\in N_T\times N_T) \quad \text{and} \quad a,b\in[num(m,n)].$$

let ω be the set of any sequences $i,i_1,\ldots,i_k,j$ for $i,j\in[num(m,n)]$ such that every pair (i,i_1), (i_k,j) and (i_p,i_{p+1}) for $p=1,\ldots,k-1$ - is an edge of the graph S.

The sequence $i, i_1, \ldots, i_k, j$ can be interpreted as the path from i to j.

Define the mapping $\tilde{\varphi}_\omega : \Lambda' \to (\Lambda'')^*$ in the following way:

$$(\forall \lambda_{ij} \in \Lambda')\tilde{\varphi}_\omega(\lambda_{ij}) = \lambda_{ii_1} \ldots \lambda_{i_k j}.$$

Example 4.3.2.1. For the structure of system S / see example 3.1.1 / the elements of an alphabet are following :

$$\{\lambda_{12}, \lambda_{21}, \lambda_{23}, \lambda_{32}\} \quad .$$

□

Let $\tilde{\Psi}_\omega : (\Sigma \cup \Lambda') \to (\Sigma \cup (\Lambda''))^*$ be the homomorphism defined as follows:

$$\tilde{\Psi}_\omega(a) = \begin{cases} \tilde{\varphi}_\omega(a) & \text{for } a \in \Lambda' \\ a & \text{elsewhere} \end{cases}$$

Now, the matrix of coprocesses should be build on the basis of the expression $\tilde{\Psi}_\omega(\overline{WR}) = \overline{\overline{WR}}$, instead of the expression $\overline{WR}$. As a result we obtain a matrix of coprocesses, which satisfies the isomorphism among G_T and S, requirements.

Example 4.3.2.2. For our an example system we can obtain:

$$\tilde{\Psi}(\overline{WR}) = \overline{\overline{WR}} = AB\lambda_{12}CD\lambda_{21}((\lambda_{12}\lambda_{23}EF\lambda_{32}\lambda_{21} \cup \lambda_{12}G\lambda_{21}B)\lambda_{12}D\lambda_{21})^*\lambda_{12}\lambda_{23}EH$$

and:

$$\overline{\overline{WR}}_{11} = AB\lambda_{12}((\lambda_{12} \cup \lambda_{12}B)\lambda_{12})^*\lambda_{12},$$

$$\overline{\overline{WR}}_{12} = CD\lambda_{21}((\lambda_{23}\lambda_{21} \cup G\lambda_{21})D\lambda_{21})^*\lambda_{23},$$

$$\overline{\overline{WR}}_{21} = (EF\lambda_{32})^*EH.$$

The modified matrix of coprocesses can be graphically expressed by graphs :

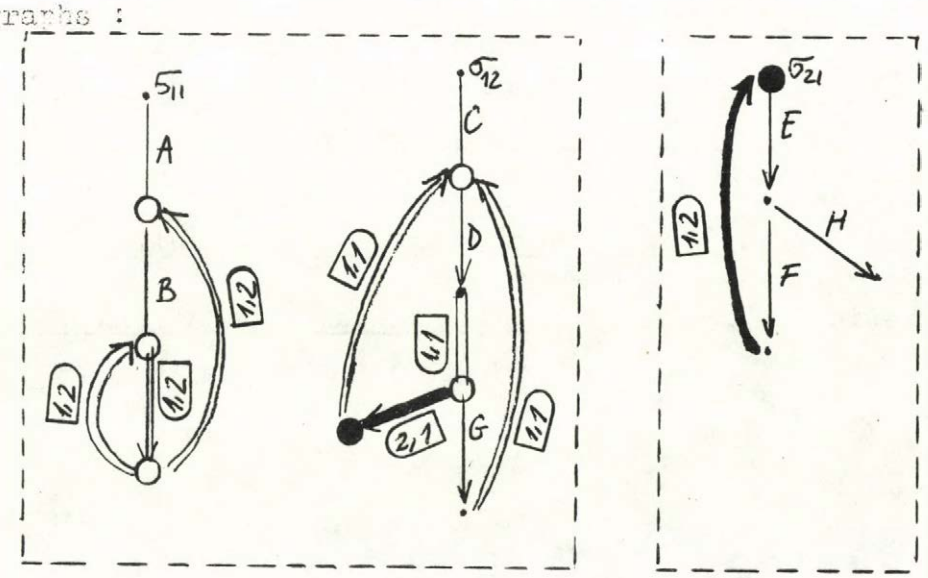
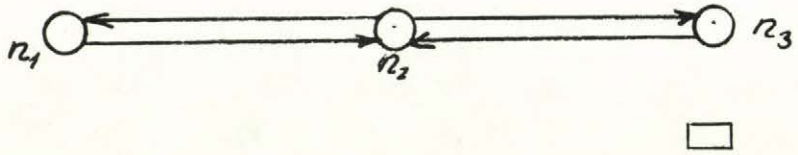


Fig 7

The graph of transmission G_T has the form:



Graphs G_T / see fig.3 / and S / see fig.1 / are isomorphic.

Acknowledegements

I would like to thank R.Janicki for his suggestions and criticism.

References.

[1] Brinch Hansen P.: Distributed processes: a concurrent programming concept. Comm. ACM, vol. 21, 11, 1978, pp.934-941.

[2] Conway M.E.: Design of a separable transition-diagram compiler. Comm.ACM, 1963, pp.396-408.

[3] Janicki R.: Vector of coroutines over Blikle nets. Lecture Notes in Computer Sciences, vol.56, 1977, pp. 113-119.

[4] Janicki R.: Outline of the vectors of coroutines theory. PWN, Warszawa-Łódź, to appear / in polish /.

[5] Janicki R., Just J.R.: Analysis of distributed computer systems by means of coroutines. Preprint of SIMULA Workshop, Budapest, 1980.

[6] Just J.R.: An algebraic model of distributed computer systems. Proc. 5-th Conf. on the Theory of Operating Systems, Visegrad, Hungary, 1979, pp.311-323, also: Postępy Cybernetyki /3/, 1, 1980.

[7] Just J.R.: Analysis and synthesis of distributed computer systems by algebraic means. Proc. 6-th Conf. on the Theory of Operating Systems, Visegrad, 1980.

[8] Mazurkiewicz A.: Proving algorithm by tail function. Information and Control, 18, 1971, pp.220-226.

[9] Mazurkiewicz A., Pawlak Z.: Mathematical Foundations of Computer Science. PWN, Warszawa, 1977.

A CODASYL modification-efficiency problem

O. Kiss - P. Radó
Computer and Automation Institute Hungarian Academy of Sciences

1. Introduction

This article is devoted to the modifications of ISDOS/DBMS. The ISDOS/DBMS is a CODASYL type data base management system consisting of FORTRAN subroutines and it was produced in 1973. This data base management system serves as the basis of system PSL/PSA and is used by it to store not very large amount of data of highly complicated structure. The modifications are intended to speed up this data base management system. To perform this task two view-points were taken into account: a/ For reasons of convenience, elegance and practice we have come to the decision, that necessary modifications should not touch the programs of system PSL/PSA even nor programs which communicate with data management system via user interface of it /this communications means subroutine calls/. b/ As the ISDOS/DBMS has general purpose we had to perform modifications independently of system PSL/PSA. Before discussing these modifications first we should like to tell something about ISDOS/DBMS.

2. ISDOS/DBMS

The ISDOS/DBMS as was mentioned, consists of FORTRAN subroutines. The common user communicates with datas in data base via a user interface. Routines contained in this

interface can be mapped more or less to DML commands of CODASYL proposal. In correspondence with CODASYL proposal DML commands receive their paramters either from user working are or they are given explicitly in commands by the user. However commands of ISDOS/DML - as they are subroutines - are operating with passing parameters according to FORTRAN language. Here we do not discuss these so called DML routines /see[2]/, later we shall deal only with the modified ones.

ISDOS/DBMS does not contain the concept of subschema. DDL possiblilities are also very limited comparing with those of CODASYL. For example there is no possibility to define REALM entry.

In the RECORD entry one can not define the LOCATION MODE of record type. Before modifications the strategy of locating records in ISDOS was the following: Records are put into pages of data base essentially in the order of arrival, sequentially. Modifications here entirely altered this strategy. For reasons of memory at one time only some pages of data base are in the memory and during the execution of the user program the not recquired pages are written on to disk the ones recquired at that time are read into memory. The paging is based on an algorithm [3].

In data subentry there is possiblility to determine data items only of INTEGER, REAL, CHARACTER and DB-KEY types, FORTRAN possibilities are followed by these types. There is also no possibility to define data items of variable length. /Though this option was taken into account by producers of system, it was built in, but ISDOS/DBMS is unable to receive such an item, there is no software to

support this facility./

In the SET entry it is not possible to determine SET MODE clause. The SET MODE in ISDOS according to the terminology of DBTG is CHAIN LINKED TO PRIOR and for each member type of a set type is valid the LINKED TO OWNER option too. Possibilities of SET ORDER option are FIRST, LAST, NEXT, PRIOR, IMMATERIAL and SORTED. For sorted sets sort can be performed only BY DEFINED KEYS /in the ISDOS the DEFINED KEYS are called SORT KEYS/. The defined key may be only one data item contained in each member type of set type, and records of set are ordered according to increasing value of sort key.

In the member subentry MANUAL and OPTIONAL are defaults for the options storage and removal class, and one can not change them. Hence if a record is stored /in ISDOS created/ it will not be connected automatically to some owner in a set type. The KEY option - which is property of set type in ISDOS, so it is not in member subentry - furthermore SEARCH KEY and SET SELECTION options are missing too.

So one can see, that ISDOS/DBMS is a vigorously restricted version of the CODASYL proposal and is not able to manage datas of large mass. Now we shall discuss other problems of retrieving records in ISDOS.

3. Sorted set phylosophy of ISDOS/DBMS

There are provided only two modes of retrieving a record from data base in ISDOS. The first possibility is performed using data base key and is a physically direct access. This possibility is quite uncomfortable, because noting data base keys for each record would cause much cost of memory or time.

The second and most applicable retrieval mode is searching for a record in a set. If the set is sorted, the record may be obtained by a single command using the value of its sort key. However this logically direct access does not mean physically direct access. Retrieving is performed by sequential search on a physical chain /pointer chain/ and ordering is used to decide /based on value of sort key/ which direction is worhly to begin search in the chain or when to give up the search. Our opinion is, that such an application of sorted set concept of CODASYL proposal is impractical. Sorted set is useful in cases when set is needed to be scanned sequentially in an order corresponding to value/s/ of certain data item/s/.

In our opinion the problem is caused by the fact, that the direct access and the concept of sorted sets are mixed up. There is no possibility to define - even on logical level not to mention the physical realization - direct access of records but for the members of sorted sets.

4. The HASHED set property and the CODASYL proposal

The CODASYL provides logically direct access for each record based on values of some data items of it if records are organized into one set. To speed up search in a set SEARCH KEY is proposed. SEARCH KEY has no affect on sequential scanning of set, direct access is quicken by organizing an index table. If the set is sorted and search key is the same as sort key access can be quicker.

Other possibility of direct access provided by CODASYL is the CALC LOCATION MODE. It means, that placement of records is computed via a hash algorithm using certain data items of them and has

not relation with the set ordering.

We shall combine these two ideas in the HASHED set property proposed by us.

The SEARCH KEY /index table/ is not sufficiently rapid and organization and maintanance of it is slightly difficult algorithm recquiring more cost in place and time. The location via CALC is fast, simple and effective, but this is a property of record type; it would be difficult to build into the system ISDOS /here we think of the compability with the user interface/.

The solution is a compromise, a hash-algorithm /CALC/ using data base key of owner of set and logical key which is a data item of member. This algorithm computes the data base key of the member record /which is the physical address in ISDOS/DBMS/. Hence HASHED is not a record property but belongs to set type. One can indicate a data item as logical key in a member record type and using it records /i.e. members/ can be selected from a set occurrence in a logically and physically effective way. This procedure is similar to SEARCH KEY.

The CALC location mode does not support sequential access of records placed using such a way.

The HASHED - as a set property - can not do so therefore scanning of members belonging to one owner is provided. The sequence of groups consisting of records of the same logical key is LIFO. It is necessary to change the physical realization of SORTED set as well to make ISDOS move effective. Members of SORTED set will be placed via a hash-algorithm using key, so the possibility of direct access will be much faster. The possibility of sequential scanning will be unchanged

For the sake of comparision we note, that in a HASHED set of SYSTEM owner the location mode of member is the same as CALC in CODASYL. One record type may be member in several set types, so we have to provide direct access to a record from several owners, however only one key may be used to access a record located via CALC.

Summarizing, the HASHED set property connects three independent concepts of CODASYL. If a set type is HASHED, then

1. its order is LIFO, more exactly it is LIFO for the groups consisting of records with the same logical key. Within a group the ordering is random.

2. The data base keys of it's members is computed using the data base key of owner of set and logical keys of members.

3. SEARCH KEY is provided for each member type of set type This SEARCH KEY can be considered as a virtual index table.

In so far as a set type is SORTED, than there is only one change, i.e. the order of set is sorted but location mode of members and search-key are unchanged.

5. The HASHED set property and the ISDOS/DML

As we have already mentioned, ISDOS/DBMS consists of subroutines written in FORTRAN. The introduction of HASHED set property recquired significant changes in data base management system as well as in DML. The reason of changes in DML is that the former record location strategy which was essentially sequential was altered.

The main problem is that the location of a new record in data

base is performed by the command routine CR /Create a Record, more exactly allocate place in data base for a record type/. At the time of CR the data base management system has no information about the contents of data items of record hence it is unable to compute data base key /physical address/, the NEAR TO OWNER strategy can not be accomplished either since the owner is not known yet. This problem is unknown for the general CODASYL, because in CODASYL there is not such a command as CR but only CRS /Create a Record and Store data - more exactly locate and place a record filled with datas/. The command CRS corresponds to the STORE of CODASYL. If the command CR would be cancelled /it is impossible for reasons of compability/ the problem would not be solved yet, because in case of a HASHED member record it is necessary to know the owner too /HASHED is a set property/, which will be turn out at the time of the appropiate AMS command /Add Member to Set, i.e. the current of a record type is connected to the current of set type/. The ISDOS/DML command AMS corresponds to the command CONNECT of CODASYL. We should like to remind, that AUTOMATIC storage class is not known in ISDOS.

The data base key phylosophy of CODASYL-78 seems to be a solution /and it is appropiate for the compability with the current version too, only the DB-KEY item type have to be cancelled/. The CODASYL-78 defines data base key as an item identifying a record during the run of a user program. This data base key has no relation with the physical location - as is written in the report [4]. A data base key of such a type means, that the system has no need to allocate data to a permanent place at the time of creating it but data may be stored in somewhat a buffer /say this buffer is addressed by data base key/ and records will be moved into data base during the run of the program at a propitious moment for

the system.

It is quite clear that we can't perform our task in such a way because of problems with memory and time. The main worry is, that the system must be able to remember the data base key given to user and based upon it must be able to find record in the buffer - or in data base if the record already has been allocated there. This means significant administration with more costs of memory and time.

We have accepted the followings as a compromise: the CR command issued by the user allocates place for the record in a buffer and the data base key returned by it is an address in the buffer. This key is valid until the first AMS issued to the record. The record is allocated to a permanent place in data base and is connected to the set according to it's order by the command AMS. Hence the main change at the user interface is that the data base key is altered by the first AMS issued to the record. Another uncomfortable consequence is that if a record is not member at least in one set occurrence it can not be set to current owner of a set type to disable the user to connect member record to it. It seems to be logical to suppose, that before the user connects a member to an owner record it has had to provide some access to the owner itself - which means in ISDOS/DBMS a set membership. If a record is not member in some set it can not be placed to the data base. This is a result of the fact, that a record is moved into the data base after the first AMS issued to it.

Further part of DML is unchanged disregarding the command FMSK /Find Member in Sorted set based on logical Key/ which was used formerly only for SORTED sets and now can be issued also for HASHED sets and provide not only logically but "almost"

physically direct access. The "almost" means that not the record but only the page containing it is determined by the hash-algorithm, within a page the search is sequential.

6. The record allocation strategy

The introduction of HASHED set has changed entirely the record allocation strategy which was in fact sequential previously. As it has become clear already, allocation of records is performed /not only for the members of a HASHED set/ not by CR but by the first AMS. After these the work of CR is restricted to allocate place in memory /unfortunately this has more cost/ and the address here will be the data base key returned. The buffer is a page of the data base with comparatively small size and 0 sequence number staying in memory during the execution of user program and the content of it never will be written into disk. The real location into the data base is the duty of AMS /this means the deletion of record from the buffer too/. This is an algorithm to be shown later for HASHED sets, for SORTED sets is essentially the same as before with the difference, that the sequence number of page where the record is to be located is computed also with the hash-algorithm. For other sets the system attempts to locate member records on the page of their owners. Let's see the hash-algorithm we want use for HASHED and SORTED sets in detail.

The hash-algorithm produces an equally distributed number between 1 and NPAGES /number of pages in data base/ using logical key and data base key of owner. This will be the sequence number of page where the record have to be located. However if the record is member in another set, i.e. it has been located in the data base using it's owner in this set,

then the second AMS issued to the record places a pointer to the computed page showing to the record. This algorithm is complicated by the fact, that it is allowed to have several records with the same logical key in a certain set. /This corresponds to the DUPLICATES ARE ALLOWED option of search key clause in CODASYL./

Some words about the problem of overflow. Each tenth page of the data base is an overflow page. No records are allocated into this page by the hash-algorithm, but if a page is full it can be continued on-this one. The treatement of overflow pages is the following: after registering that there is no more room on the page the system looks for a free overflow page using some algorithm. Hence during retrieval the page then it's overflow pages are examined by the data base management system. The overflow page also can be continued resulting overflow chains.

REFERENCES

[1] T. William Olle, The CODASYL approach to data base management John Wiley & Sons /1978/

[2] E.A. Hershey, P.W. Messink, A data base management system for PSA based on DBTG71, ISDOS Working Paper No. 88 /1975/

[3] E.A.Hershey, R.L. Dissen, P.W. Messink, A description of ADBMS, ISDOS Working Paper No. 122 /1975/

[4] Report of the CODASYL DDL Committee, Information Systems Vol. 3. N.4 /1978/

Summary

This paper deals with a modification of a CODASYL type data base management system called ISDOS/DBMS. This system is based on the widely known GPLAN, and is used by PSL/PSA to store not very large amount of data of highly complicated structure.

Having installed PSL/PSA for ES computers, efficiency problems occured, which made a strong restriction for the possible applications. Since the data management may be considered as quite an independent part of the whole system, the solution of the problem happened to be a modification of the ISDOS/DBMS only.

Taking into account the special purpose of the system, an unconventional feature was introduced. The HASHED record placement strategy is a facility between the standard CODASYL CALC location mode and indexing. Although the concept makes the physical reorganisation of the data base more difficult, it provides a more flexible method to organise an effective direct retrieval than CALC. The advantegous and disadvantegous features of the proposal are discussed independently of the PSL/PSA application.

В.А. Крюков, Э.З. Любимский, М.Р. Шура-Бура

Управляемая виртуальная память.

Стоимость оперативных запоминающих устройств не позволяет в настоящее время полностью удовлетворить потребности задач в оперативной памяти. Это вынуждает программиста располагать часть своих рабочих массивов во вспомогательной памяти (на магнитных барабанах, дисках или лентах). Поскольку доступ к вспомогательной памяти возможен только через оперативную, то для работы с такими "нерезидентными" массивами программист должен выделить в оперативной памяти специальные буфера и организовать их своевременную перезагрузку. Для обеспечения такой перезагрузки приходится переорганизовывать программу - её работа квантуется в соответствии с размерами буферов. При этом в программе вместо обращений к нерезидентным массивам пишутся обращения к соответствующим буферам. Кроме того, приходится производить преобразование индексов таких массивов. Особые трудности приходится преодолевать в тех случаях, когда для работы с одним массивом требуется использовать несколько буферов, информация в которых может пересекаться. Все вышесказанное значительно усложняет разработку программ.

Появление виртуальной памяти избавило программиста от этих трудностей, поскольку теперь многоуровневая память ЭВМ представлена ему как логически однородная одноуровневая память. Схему организации виртуальной памяти можно кратко описать следующим образом. Вся используемая задачей виртуальная память отображается частично на оперативную, а частично на вспомогательную память машины. Информация об этом отображении находится в памяти ЭВМ и доступна аппаратуре преобразования виртуального адреса. Каждый поступивший в уст-

ройство управления виртуальный адрес заменяется на адрес оперативной памяти, если соответствующая часть виртуальной памяти отображена на оперативную память. Если же эта часть виртуальной памяти отображена на вспомогательную память, то возбуждается специальная процедура операционной системы, которая обеспечивает загрузку в оперативную память требуемой информации. Как правило, для этого приходится освободить в оперативной памяти место, переписав часть находящейся там информации во вспомогательную память. Поскольку операционная система не знает характер последующей работы с массивами в задаче, то она не может минимизировать количество обменов между уровнями памяти так же удачно, как это может сделать программист при явном использовании многоуровневой памяти. Поэтому платой за удобство виртуальной памяти является потеря эффективности.

Для того, чтобы объединить удобство виртуальной памяти с эффективностью явного использования многоуровневой памяти, предлагается метод управляемой виртуальной памяти (УВП), идея которого заключается в следующем.

Программист пишет программу в расчете на виртуальную память, но получает возможность с помощью специальных оптимизирующих указаний управлять перемещением информации между уровнями памяти. При этом программисту предлагается широкий спектр возможностей для управления. Он может задавать лишь общие закономерности работы с массивами, а может в процессе счета давать исчерпывающие указания о том, какую информацию можно изгнать из оперативной памяти, а какую поместить на освободившееся место. Эти исчерпывающие указания обеспечат программе такую же эффективность, как и при явном использовании многоуровневой памяти (и при этом метод УВП гораздо удобнее). При полном же отсутствии в задаче оптимизирующих указаний метод УВП сводится

к обычному методу виртуальной памяти.

Программисту предлагаются следующие возможности управления виртуальной памятью. Он может сообщить операционной системе, какие массивы следует поместить целиком в оперативную память (такие массивы называются резидентными). Остальные массивы располагаются во вспомогательной памяти и называются нерезидентными. Для работы с ними программист может выделить в оперативной памяти буфера, называемые "зонами видимости" или просто "зонами". Настройкой зон (сменой информации в них) можно управлять непосредственно или косвенно.

Непосредственное управление заключается в том, что в процессе решения задачи с помощью специальных операторов настройки операционной системе сообщается диапазон элементов массива, наличие которых в зоне она должна обеспечить. Перенастройка зоны производится только в том случае, когда в ней отсутствуют какие-либо элементы из указанного диапазона.

Косвенное управление состоит в том, что уже при выделении для массива зоны программист задает закон, по которому операционная система должна осуществлять перенастройку этой зоны. Операторы настройки в таком случае уже не нужны. Среди способов косвенного управления следует отметить синхронную и сегментную настройки.

Как правило, некоторые массивы в задаче обрабатываются синхронно, т.е. относительно одновременно используются элементы этих массивов с одинаковыми индексами. При выделении зон для таких массивов программист может указать на их синхронность. Тогда при перенастройке одной из таких синхронных зон операционная система будет соответствующим образом перенастраивать и остальные. Сегментная настройка применяется тогда, когда при решении задачи в массиве относительно одновременно используется только сегмент определенной

длины, который в процессе счета продвигается по массиву в том или ином направлении. При выделении зоны для такого массива программист может задать некоторую информацию об этом сегменте, которая позволит операционной системе обеспечивать при перенастройке зоны наличие в ней нужного сегмента.

Реализация метода УВП для машин, аппаратура которых рассчитана на виртуальную память, принципиальных затруднений не вызывает. Большим достоинством принципа управления виртуальной памятью является то, что он позволяет достаточно эффективно реализовать виртуальную память и на машинах с неполной виртуальной адресацией (т.е. машинах, аппаратура которых не рассчитана на виртуальную память). Именно такие машины в основном используются в настоящее время (в их числе – БЭСМ-4, БЭСМ-6, ЕС ЭВМ Ряд-I).

На этих машинах можно организовать работу с виртуальной памятью в алгоритмических языках. В этом случае выполнение функций аппаратуры преобразования виртуального адреса должен обеспечить транслятор с этого языка, т.е. он должен запрограммировать: проверку наличия в оперативной памяти требуемой информации, замену виртуального адреса на соответствующий адрес в оперативной памяти, обращение к операционной системе за вызовом в оперативную память нужной информации. Такое программное преобразование виртуального адреса связано с большими накладными расходами. Это подтверждают результаты известных работ (например, транслятор ТА-2 для ЭВМ М-20). Однако, при использовании УВП наличие в программе оптимизирующих указаний не только помогает операционной системе минимизировать количество обменов между уровнями памяти, но и позволяет транслятору при некоторых условиях значительно повысить эффективность программного преобразования виртуального адреса. Для этого программист должен не только выделить для массива зону, но и с помощью управ-

ляющих указаний гарантировать наличие в ней требуемых элементов массива (например, с помощью непосредственной или синхронной ностройки). В таком случае при обращении к элементу массива можно не производить проверку наличия этого элемента в оперативной памяти. Поскольку в зонах находится информация с последовательными виртуальными адресами, то для программного преобразования виртуального адреса выгодно использовать следующий способ.

С каждой зоной связывается пара граничных значений (АМАКС и АМИН) и некоторая переменная база (БАЗА). При каждой перенастройке зоны значение АМИН устанавливается равным виртуальному адресу первого элемента в зоне, а значение АМАКС – виртуальному адресу последнего элемента. База зоны при этом полагается равной разности между адресом в оперативной памяти первого элемента в зоне и его относительным адресом в массиве. Для проверки наличия в зоне требуемого элемента массива достаточно определить, попадает ли его виртуальный адрес в диапазон, задаваемый значениями АМИН и АМАКС. Если элемент отсутствует в зоне, то требуется возбудить процедуру, которая осуществляет перенастройку зон. Если же требуемый элемент находится в зоне, то его адрес может быть получен путем сложения его относительного адреса в массиве со значением базы зоны, т.е.

$$АОП = АОТН + БАЗА$$

где: АОП – адрес элемента в оперативной памяти,

АОТН – его относительный адрес в массиве,

БАЗА – равняется разности АОПI–АОТНI адреса в оперативной памяти и относительного адреса для первого элемента в зоне.

Если адрес оперативной памяти на данной вычислительной машине состоит из m двоичных разрядов, то верно такое равенство:

$$АОП = \left| |АОТН|_{2^m} + |БАЗА|_{2^m} \right|_{2^m}$$

т.е. можно складывать по модулю 2^m только m младших разрядов, а такое сложение можно выполнять с помощью операций базирования или модификации адреса, которые имеются в системе команд любой машины.

Таким образом, при отказе от проверки наличия элемента в зоне преобразование виртуального адреса требует не более одной дополнительно команды.

Сложнее обстоит дело с программированием работы с формальными параметрами в процедурах, которым могут соответствовать в качестве фактических параметров нерезидентные массивы (поскольку неизвестно, какие граничные значения и базу нужно использовать).

Можно использовать способ программирования формальных массивов, основанный на том, что при каждом формальном массиве программист может указать номер некоторой зоны, которая при обращении к процедуре настраивается на фактический массив. В таком случае, как правило, работа с формальными массивами выполняется не менее эффективно, чем работа с фактическими нерезидентными массивами.

Поскольку при реализации УВП для машин с неполной виртуальной адресацией потребовалось предъявить программисту дополнительные требования, то необходимо было предусмотреть средства, которые облегчали бы ему разработку программ. Следовало также обеспечить такой режим решения задачи (пусть даже ценою потери эффективности), когда управляющие указания остаются только оптимизирующими (т.е. не влияют на правильность работы программы). При решении задачи в таком режиме необходимо было обеспечить выдачу программисту информации об обращениях к отсутствующим в оперативной памяти элементам, которые не позволяют отказаться от проверки наличия в зоне требуемых элементов. Нужно было также позволить программисту постепенно и по частям оптимизировать свою программу, переходя от такого отладочного режима ре-

шения задачи к рабочему режиму.

Разработанные принципы организации УВП для машин с неполной виртуальной адресацией были реализованы на БЭСМ-6 для программ на языках АЛГОЛ и ФОРТРАН, что позволило использовать в них всю увеличенную оперативную память БЭСМ-6, которая до этого была доступна задаче лишь частично (I5-разрядный адрес позволял задаче обращаться лишь к 32 страницам памяти из I28).

Операционная система ОС ИПМ модифицирована таким образом, чтобы задача могла использовать оставшуюся оперативную память в качестве быстрой вспомогательной памяти (обмен с ней в I00 раз быстрее обмена с МБ, поскольку при этом обмене не происходит физического копирования информации, а лишь изменяются аппаратные регистры приписки - регистры соответствия между номерами математических и физических страниц памяти).

Общая схема реализации УВП для БЭСМ-6 выглядит следующим образом. На этапе трансляции в тексте программы на входном языке выделяются конструкции, относящиеся к работе с УВП, и отображаются в соответствующие таблицы и команды готовой программы. Некоторые из таких конструкций переводятся в обращения к специальной административной системе, которая помещается на время решения задачи в её оперативную память и обеспечивает динамическую работу с виртуальной памятью.

Кроме использования увеличенной оперативной памяти реализация УВП на БЭСМ-6 преследовала еще одну цель - позволить заранее готовить и выполнять на БЭСМ-6 программы для машин с большим объемом прямоадресуемой оперативной памяти (например, крупных моделей ЕС ЭВМ).

Для достижения этой цели все управляющие указания оформляются в виде комментариев или операторов над некоторым фиктивным массивом.

Реализованный в I976г. первый вариант системы УВП позволял ис-

пользовать в задачах до 100000 слов виртуальной памяти, которая отображалась на два уровня памяти – оперативную и быструю вспомогательную память (в качестве которой использовалась оставшаяся часть увеличенной оперативной памяти БЭСМ-6). В 1978г. был реализован второй вариант системы УВП, позволивший программисту прямоадресовать до $4 \cdot 10^6$ слов виртуальной памяти (что соответствует максимальному размеру виртуальной памяти в ЕС ЭВМ – 16 мегабайт). В этом варианте виртуальная память отображается на три уровня памяти – оперативную, быструю вспомогательную и память на магнитных барабанах или дисках.

Литература

I. Belady, L.A.
A study of Replacement Algorithms for a Virtual Storage Computer. IBM Systems Journal, vol. 5, No.2, I966.

2. Denning, P.J.
Virtual Memory.
Computing Surveys, Vol. 2, No. 3,
September I970.

3. И.А. Бахарев, Н.А. Коновалов, В.А. Крюков, Э.З. Любимский, В.В. Мартынюк, Е.В. Хухлаев.
Методика использования расширенной памяти БЭСМ-6. Препринт, ИПМ АН СССР, I977г.

4. Н.А. Коновалов, В.А. Крюков, Э.З. Любимский
Управляемая виртуальная память. Москва, Программирование, I977г. № I.

5. С. Мэдник, Дж. Донован.
Операционные системы. Москва, Мир, I978г.

A decision problem related to a discrete doubly stochastic process

Krámli A., Lukács P., Vassel R.

Budapest

The authors have investigated many different /discrete or continuous time parameter/ point processes to solve stochastic optimization problems related to the work of computer systems, such as optimal paging algorithms, [1] stochastic analysis of recovery procedures providing data bases [2] etc..

The process studied in the present talk, and its continuous time parameter analogue are widely used in the literature ([3], [4]) - we have chosen it for the purposes mentioned above. Our contributions to the well known results are the proof of an ergodic property of the decision process and numerical computation of the stationary distribution function of it.

Let $\{\zeta_n\}$ be a Markov chain with two states 0 and 1 - $P\{\zeta_n = 1 \mid \zeta_{n-1} = 0\} = P\{\zeta_n = 0 \mid \zeta_{n-1} = 1\} = P.$

The observable process $\{\xi_t\}$ having also the states 0 and 1, is governed by $\{\zeta_n\}$ as follows

$$P\{\xi_n = 1 \mid \zeta_n = 0, \zeta_{n-1}, \dots, \zeta_0, \xi_{n-1}, \dots, \xi_0\} = p_0$$

$$P\{\xi_n = 1 \mid \zeta_n = 1, \zeta_{n-1}, \dots, \zeta_0, \xi_{n-1}, \dots, \xi_0\} = p_1$$

Notice that the pair $\{\zeta_k, \xi_k\}$ is a Markov-chain, but $\{\xi_k\}$ is not Markovian.

Let π_0 be the prior probability of the event $\{\zeta_0 = 0\}$ and $\pi_n = P\{\zeta_n = 0 \mid \xi_n, \dots, \xi_0, \pi_0\}$

It can easily be proved (see [3]) that π_n is a sufficient statistics for the value of ζ_n, more precisely

$$(1) \quad P(\xi_{n+1}, \dots, \xi_{n+k}, \zeta_n, \dots, \zeta_{n+k} \mid \pi_n, \xi_n, \dots, \xi_0, \pi_0) = P(\xi_{n+1}, \dots, \xi_{n+k}, \zeta_n, \dots, \zeta_{n+k} \mid \pi_n)$$

It follows from relation (1) that the process $\{\pi_n\}$ is a discrete time parameter Markov chain with continuous state space. From Bayes' theorem we get

$$(2) \quad \pi_n = f_0(\pi_{n-1}) =: f_0(p_0, p_1, \pi_{n-1}) = \left(1 + \frac{(1-p_1)(P\pi_{n-1} + (1-P)(1-\pi_{n-1}))}{(1-p_0)((1-P)\pi_{n-1} + P(1-\pi_{n-1}))}\right)^{-1}$$

if $\xi_n = 0$

and

$$(2') \quad \pi_n = f_1(\pi_{n-1}) = f(1-p_0,\ 1-p_1,\ \pi_{n-1})$$

if $\xi_n = 1$.

For fixed π_{n-1}

$$(3) \quad P\{\pi_n = f_0(\pi_{n-1})\} =: g(\pi_{n-1}) =$$

$$= (1-p_0)\left((1-P)\pi_{n-1} + P(1-\pi_{n-1})\right) +$$

$$+ (1-p_1)\left((1-P)(1-\pi_{n-1}) + P\pi_{n-1}\right)$$

$$(3') \quad P\{\pi_n = f_1(\pi_{n-1})\} = 1 - g(\pi_{n-1})$$

If π_{n-1} and π_n have the probability distribution functions $H^{(n-1)}(y)$ and $H^{(n)}(y)$ then

$$(4) \quad H^{(n)}(y) = \mathbb{A}(H^{(n-1)})(y) =: H^{(n-1)}(g_0(y))\, g(g_0(y)) -$$

$$- \int_0^{g_0(y)} g'(x)\, H^{(n-1)}(x)\, dx +$$

$$+ H^{(n-1)}(g_1(y))(1 - g(g_1(y))) +$$

$$+ \int_0^{g_1(y)} g'(x)\, H^{(n-1)}(x)\, dx, \text{ where } g_i(y) := f_i^{-1}(y),\ i = 0,1.$$

The equality (4) can be derived from equalities (3) and (3') and from the rule of "integration by parts" valid also for Stieltjes integrals. A stationary distribution must be a solution of equation

$$H(y) = \mathbb{A}(H)(y)$$

The problem of existence and uniqueness is open. The numerical experiments, the simulation /see tables 1 and 2 / and analytic considerations show that $H'(y)$ may have infinitely many singularities. The process π_n is obviously non-ergodic but we can prove the following stability theorem.

Theorem: For a given realization $\{\xi_k\}$ let $\{\pi_{n,x}\}$ and $\{\pi_{n,y}\}$ be two sequences generated by the consecutive applications of transformations (2), (2') assuming $\pi_{0,x} = x$ and $\pi_{0,y} = y$.

Then there exists a constant C such that for almost every realization $\{\xi_k\}$

$$|\pi_{n,x} - \pi_{n,y}| < e^{-Cn + o(n)}$$

Proof. Let σ_n be equal to $1 - \pi_n$. Then relations (2) and (2') turn into a vector-matrix form:

$$(5)\quad \lambda_1 \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix} \begin{pmatrix} 1-p_0 & 0 \\ 0 & 1-p_1 \end{pmatrix} \begin{pmatrix} \pi_{n-1} \\ \sigma_{n-1} \end{pmatrix} = \begin{pmatrix} \pi_n \\ \sigma_n \end{pmatrix} \quad \text{if } \xi_n = 0$$

$$(5')\quad \lambda_2 \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix} \begin{pmatrix} p_0 & 0 \\ 0 & p_1 \end{pmatrix} \begin{pmatrix} \pi_{n-1} \\ \sigma_{n-1} \end{pmatrix} = \begin{pmatrix} \pi_n \\ \sigma_n \end{pmatrix} \quad \text{if } \xi_n = 1$$

where λ_1 and λ_2 are appropriate norming coefficients. It can be proved that there exists a constant $\varepsilon > 0$ such that $\varepsilon < \pi_{n,x} < 1-\varepsilon$, $\varepsilon < \pi_{n,y} < 1-\varepsilon$ for every n=1,2, . . . Thus, if we ignore the norming coefficients, then it is sufficient to prove that

$$(6) \quad \left| \frac{\pi_{n,x}}{\sigma_{n,x}} - \frac{\pi_{n,y}}{\sigma_{n,y}} \right| < e^{-Cu+\sigma(n)}$$

For every realization $\{\xi_k\}$ the consecutive applications of (5) and (5') results in a product of n matrices of type

$$\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}\begin{pmatrix} 1-p_0 & 0 \\ 0 & 1-p_1 \end{pmatrix} \quad \text{resp.} \quad \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}\begin{pmatrix} p_0 & 0 \\ 0 & p_1 \end{pmatrix}.$$

By choosing appropriately the norming coefficients we can reach that the determinant of this product is equal to 1 for every n. By elementary calculation we can derive that the elements of this product tends exponentially to ∞ if the law of large numbers is valid for the realization $\{\xi_k\}$. /Notice that the process $\{\xi_k\}$ satisfies the strong mixing condition, so the law of large number is valid almost surely./ From this last statement we get (6).

Numerical results

The values of the parameters: $P = 0.01$, $P_0 = 0.8$, $P_1 = 0.2$.

The "stationary density function" obtained by Monte Carlo method. /One realization of length looooo./ Table 1. containes the values:

$$100 \cdot \left(H\left(\frac{i}{100}\right) - H\left(\frac{i-1}{100}\right) \right), \quad i = 1, \ldots, 100.$$

i=	1	2	3	4	5
	0.	25.3493	5.6401	1.4430	0.7000
6	0.2910	4.3790	1.3700	0.6030	0.2140
11	0.8730	0.3560	0.2650	0.0920	0.2020
16	0.1570	0.0940	0.1270	0.0370	0.2230
21	0.0190	0.6540	0.1600	0.2760	0.3060
26	0.1060	0.1270	0.1130	0.0390	0.0520
31	0.0410	0.2260	0.1130	0.1470	0.1140
36	0.1200	0.0500	0.0400	0.0230	0.0570
41	0.0360	0.0380	0.0450	0.0770	0.1250
46	0.0560	0.0660	0.1530	0.3130	0.0600
51	0.0710	0.3120	0.1490	0.0360	0.0490
56	0.1170	0.1010	0.0550	0.0470	0.0790
61	0.0070	0.0270	0.0540	0.0620	0.1330
66	0.1130	0.1510	0.1170	0.2330	0.0390
71	0.0370	0.0300	0.1170	0.1150	0.1050
76	0.2790	0.2560	0.2040	0.0710	0.0000
81	0.2390	0.0830	0.1150	0.1020	0.1000
86	0.2360	0.1250	0.2790	0.0510	0.2360
91	0.2040	0.0090	1.9490	5.0201	0.2900
96	0.7320	1.4310	6.0691	27.8955	0.

Table 1.

The "stationary density function" obtained after 100 iteration of the equation

$$H_n(x) = \mathbb{A}\left(H_{n-1}(x)\right)$$

storting from the distribution function

$$H_0(x) = \begin{cases} 0 & \text{if} \quad x < 0 \\ x & \text{if} \quad 0 \leq x < 1 \\ 1 & \text{if} \quad x \geq 1 \end{cases}$$

The content of Table 2 is analogous to the Table 1.

21.9350	6.3126	2.0733	1.1670	0.6732
4.3132	2.4030	0.8533	0.5544	1.0412
0.7697	0.4637	0.2177	0.2514	0.2590
0.2004	0.1645	0.2030	0.2607	0.1003
0.1813	1.0726	0.1990	0.1204	0.5335
0.1457	0.1185	0.2075	0.0584	0.0354
0.0931	0.2078	0.2435	0.1211	0.2923
0.0602	0.1343	0.0490	0.0623	0.1097
0.0789	0.0909	0.1523	0.1541	0.0751
0.0741	0.2751	0.2420	0.0756	0.0436
0.0480	0.0756	0.2420	0.2751	0.0741
0.0751	0.1541	0.1624	0.0909	0.0789
0.1099	0.0623	0.0490	0.1343	0.0601
0.2923	0.1211	0.2435	0.2078	0.0931
0.0354	0.0584	0.2075	0.1185	0.1457
0.5335	0.1204	0.1990	1.0727	0.1813
0.1003	0.2607	0.2030	0.1645	0.2004
0.2590	0.2513	0.2177	0.4637	0.7697
1.0412	0.3544	0.8533	2.4030	4.3132
0.6732	1.1670	2.0733	6.3124	21.9345

Table 2.

References

[1] Benczúr. A., Krámli, A. and Pergel, J., A Bayesian approach to optimal performance of page storage hierarchies, Acta Cybernetica 3, 79-89 /1976/

[2] Benczúr, A., and Krámli, A., A note on data base integrity, Acta Cybernetica 4, 181-186 /1977/

[3] Girshik, M.A. and Rubin, H., A Bayes approach to a quality control model, Ann. Math. Statist. 23, 114-125 /1952/

[4] Rudemo, M., Doubly stochastic Poisson processes and process control , Adv. Appl. Prob. 4, 318-338 /1972/

AN EXPERIMENTAL FACILITY FOR THE ANALYSIS OF PARALLELISM IN DATA BASES

Alain KURINCKX
IRIA-LABORIA
BP.105
F 78150 Le Chesnay
FRANCE

1. Introduction

One of the main trends in computer architecture is the generalization of small and inexpensive machines built around microprocessors (microcomputers or personal computers). Due to their basic limitations it will become necessary that such machines co-operate to be able to support practical large-scale applications. This is one of the reasons why in many research centers, studies are carried out in the area of distributed computer architecture.

The purpose of this paper is to describe one of the developments in this direction undertaken in the "performance evaluation" group at IRIA. We have chosen to examine first, problems arising from data bases implementation, both from a theoretical point of view and from a practical one, on such a distributed architecture. Therefore we are designing and building an experimental facility in which logical and quantitative investigations will be carried out.

Distributed data bases (DDB) are good examples of distributed systems for the following reasons. First, the operations on a multi-users DDB are, by nature, parallel : transactions can be partially served simultaneously on each processor. Secondly it is necessary to distribute the large amount of data on several machines. At last, there are several motivations for storing copies of the same information at more than one center. First responses to queries can be made more quickly if the information requested is stored at the center where the query is received. Secondly, the breackdown of certain center will not cause everyone to less access to the information if copies are stored at several centers.

Before to review the main problems arising from the use of DDB, we have to note the importante of computer networks (both public and local) which permit to connect easily the machines.

2. Some fundamental problems

We have just seen why it is necessary to distribute data bases on a multiprocessor architecture. However, the distribution leads to an increase of the amount of communication and the benefit obtained from the parallelism may be lost in the synchronization delays and in managing the exchange of messages. So a first problem is to determine the degree of decentralization which yields optimum performance; some preliminary results concerning this point can be found in [CGP. 79]. There are two consequences of access parallelism on the communications.

First, access conflicts arise as soon as several users share the information. A first solution is to lock all the data needed by a transaction during the whole update sequence. An alternative is to lock the data as the update goes on (with a non-zero probability of deadlocks). But in all cases, lock

mechanisms require exchange of messages.

The second consequence of the parallelism is due to the necessity of having copies of the same information at several centers. If updates to a multiple copy DDB originate at various centers, several major problems arise. Each center becomes aware of update originating at other centers only after some unpredictable delay. Thus, an user may make an update request based on information currently available at his center although the information has already been modified by another user at a different center. If not controlled, such situations can cause the internal consistency constraints of the data base to be violated even if each update maintains consistency when applied alone. In single copy data bases, mutual exclusion can be provided by lock mechanisms, but in the multiple copy context the volume and duration of the communications required to set global locks, is often prohibitive. It will be therefore necessary to stop parallelism during certain periods to insure consistency.

A variety of techniques for concurrent updates have been proposed recently [Ell.77, Tho.76, Lam.78, LeL.77, GeS.78] but there are very few studies which lead to the performance of the proposed algorithms. In this area we can quote [GeS.78, Pla.78]. In [KlP.79], the authors examine how the use of a single channel as the communication medium can provide a solution to the problem of maintaining consistency. A model of a DDB is developed to study the optimization of performance as functions of the redundancy and partition of the data.

3. Description of the experimental facility

The experimental system is built with three (in its present version) microcomputers interconnected by a local network. The characteristics of the network, called "TABLE-RONDE" are the following

- a request and update language
- a manager which control the data base and execute the elementary requests or updates generated by the language interpreter.

The data are structured into pages (of 512 bytes) which are loaded into memory on request according to a mechanism similar to the virtual memory policy. Pages are allocated to relations and are stored on disks (one file per relation).

The global experimentation is subdivided into several parts of increasing complexity. In a first step, we implement a centralized, multi-users relational data base. One machine possesses all the data and the manager. The other machines have each an user in charge and send the compiled requests to the first one. The aim of this experiment is to determine the optimal degree of request decomposition (from global to very elementary requests). With this system, it will be also possible to study different policies to treat the access conflicts.

In a second step, the data will be distributed to study the effects of parallelism and the consequences of communication increase. Lastly we shall implement the most general case of DDB with multiple copy. Using this system, it will be possible to compare the performance of different algorithms devoted to insure the consistency.

4. References

[CGP.79] Coffman E.G., Gelenbe E., Plateau B., "Optimization of the number of copies in a distributed data base", Research Report n° 34, LRI (Orsay), April 1979.

[Ell.77] Ellis C.A., "A robust Algorithm for Updating duplicate Data Base", Berkeley Workshop on distributed data management and Computer network, 1977.

[GeS.78] Gelenbe E., Sevcik K, "Analysis of Update synchronization for Multiple Copy Data Bases", Research Report n° 16, LRI (Orsay), June 1978, to appear in IEEE Trans. on Computers.

[KlP.79] Klawe M., Pujolle G., "A synchronizing Technique in a Distributed System and its Performance", to appear.

[Lam.78] Lamport L., "Time, Clocks and the Ordering of Events in a Distributed System", CACM 21, 7, July 1978.

[LeL.77] LeLann G., "Distributed Systems. Towards a formal Approach", IFIP Congress 1977.

[Pla.78] Plateau B., "Evaluation des performances d'un algorithme de contrôle de cohérence d'une base de données répartie", to appear in Acta Informatica.

[Tho.76] Thomas R.H., A solution to the Update Problem for Multiple Copy Data Bases", BBN Report n° 3340, July 1976.

A PROBABILITY MODEL FOR MULTIPROCESSOR SYSTEMS

L.Lakatos/Budapest/ and T.A.Annayev/Ashabad/

1.We consider a system containing two processors with bulk arrivals of Poisson type and exponentially distributed service time. According to such assumptions for a small time interval we can have changes of the following types: 1.there occurs a group consisting of a certain number of requests /jobs/, 2.there can be finished the service of one or two jobs. We formulate the corresponding probability problem.

Let ξ_t , $t \geq 0$ be a homogeneous Markov chain with possible states $0, 1, 2, \ldots$ and with transition probabilities for t / $t \to 0$ /:

$$
\begin{aligned}
&P\{k \xrightarrow{t} k-2+r\} = \delta_{2r} + a_r t + o(t), \\
(1)\quad &P\{1 \xrightarrow{t} r\} = \delta_{1r} + b_r t + o(t), \\
&P\{0 \xrightarrow{t} r\} = \delta_{0r} + c_r t + o(t),
\end{aligned}
$$

$$k \geq 2, \; r \geq 0,$$

here δ_{ij} - Kronecker's symbol, $a_2 < 0$, $a_0 > 0$, $b_1 < 0$, $c_0 < 0$, the other a_r , b_r , c_r are nonnegative and

$$(2) \qquad \sum_{r=0}^{\infty} a_r = \sum_{r=0}^{\infty} b_r = \sum_{r=0}^{\infty} c_r = 0.$$

ξ_t can be interpreted as the number of requests /jobs/ in this system. There can be serviced one or two requests under the following conditions:
1/ if $\xi_t = k \geq 2$, then $a_0\Delta + o(\Delta)$ / $a_1\Delta + o(\Delta)$ / is the probability both requests /one of the requests/ will be

serviced, $a_r\Delta+o(\Delta)$ – the probability of the event to the moment $t+\Delta$ to occur a group consisting of $r-2$ requests;
2/ if $\xi_t=1$ then $b_0\Delta+o(\Delta)$ - the probability of event to the moment $t+\Delta$ the only request in the system will be serviced, and $b_r\Delta+o(\Delta)$, $r>1$ - the probability of the event to the moment $t+\Delta$ in the system to occur a group consisting of $r-1$ requests;
3/ if $\xi_t=0$, then for $(t,t+\Delta)$ in the system there can occur a group consisting of r requests with probability $c_r\Delta+o(\Delta)$, $r>0$.

Our purpose is to find the transition probabilities $P_{lk}(t)$ for the chain ξ_t and to investigate them as $t\to\infty$.

2. According to (1) the direct system of Kolmogorov differential equations for $P_{lk}(t)$ is of the next form:

$$(3)\quad \frac{dP_{lk}(t)}{dt}=P_{l0}(t)c_k+P_{l1}(t)b_k+\sum_{r=2}^{k+2}P_{lr}(t)a_{k-r+2},$$

$$P_{lk}(0)=\delta_{lk}, \quad k\geq 0, \qquad / \ l\geq 0 \text{ and is fixed/}$$

We introduce the following generating functions and Laplace transforms:

$$P_l(t,\theta)=\sum_{k=0}^{\infty}P_{lk}(t)\theta^k, \qquad a(\theta)=\frac{1}{\theta^2}\sum_{k=0}^{\infty}a_k\theta^k,$$

$$(4)\quad b(\theta)=\frac{1}{\theta}\sum_{k=0}^{\infty}b_k\theta^k, \qquad c(\theta)=\sum_{k=0}^{\infty}c_k\theta^k,$$

$$\tilde{P}_{lk}(s)=\int_0^{\infty}e^{-st}P_{lk}(t)dt, \qquad \tilde{P}_l(s,\theta)=\int_0^{\infty}e^{-st}P_l(t,\theta)dt,$$

$$0<|\theta|\leq 1, \quad s>0.$$

Using (3) we get

$$\frac{\partial P_\ell(t,\theta)}{\partial t} = P_{\ell 0}(t)c(\theta) + P_{\ell 1}(t)\theta b(\theta) +$$

$$+\left[P_\ell(t,\theta) - P_{\ell 0}(t) - P_{\ell 1}(t)\theta\right]a(\theta),$$

or

$$\left[s-a(\theta)\right]\tilde{P}_\ell(s,\theta) =$$

(5)

$$= \theta^\ell - \left[a(\theta)-c(\theta)\right]\tilde{P}_{\ell 0}(s) - \left[a(\theta)-b(\theta)\right]\theta\tilde{P}_{\ell 1}(s).$$

At each $s>0$ $\tilde{P}_\ell(s,\theta)$ is finite in the circle $|\theta|<1$. So to find $\tilde{P}_{\ell 0}(s)$ and $\tilde{P}_{\ell 1}(s)$ it is natural to investigate the roots of equation

(6) $$s-a(\theta)=0$$

in the circle $|\theta|<1$.

<u>Theorem 1.</u> In $[0,\infty)$ exist continuous monotone functions $\lambda_0(s)$ and $\lambda_1(s)$ for which the following relations hold

$$\lim_{s\to\infty}\lambda_0(s) = \lim_{s\to\infty}\lambda_1(s) = 0,$$

$$\lambda_0(0) = \lambda_0 \in (0,1], \quad \lambda_1(0) = \lambda_1 \in [-1,0),$$

$$\lambda_0(s) + \lambda_1(s) \geq 0, \quad s\geq 0,$$

$$a(\lambda_0(s)) = a(\lambda_1(s)) = s, \quad s\geq 0.$$

Furthermore, $\lambda_0 = 1$ then and only then if $a'(1)\leq 0$, and

$\lambda_1 = -1$ then and only then if $\lambda_0 = 1$ and $a_1 = a_3 = \ldots = 0$

Proof. We have / $s > 0$ /

$$\Theta^2 \left[s - a(\Theta) \right] = (s - a_2)\Theta^2 - \sum_{k \neq 2} a_k \Theta^k$$

and on the circle $|\Theta| = 1$

$$(7) \qquad \left| (s - a_2)\Theta^2 \right| = s - a_2 = s + \sum_{k \neq 2} a_k > \left| \sum_{k \neq 2} a_k \Theta^k \right|.$$

From this inequality and Rouché's theorem follows the equation (6) has no more than two different roots in the circle $|\Theta| < 1$ at each fixed $s > 0$. As in $(0,1]$

$$a(+0) = +\infty, \quad a(1) = 0, \quad a''(\Theta) > 0,$$

so $a(\Theta)$ has the next form

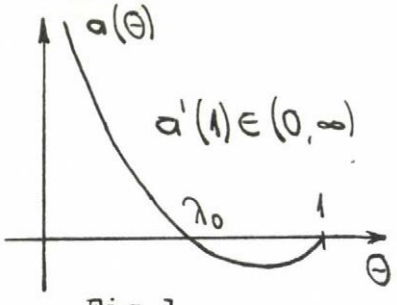


Fig.1.

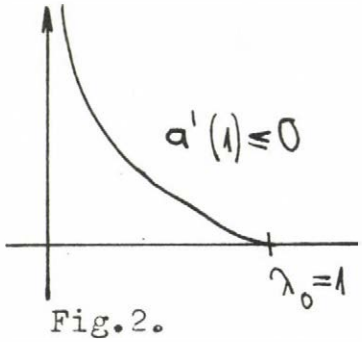


Fig.2.

The function inverse to $a(\Theta)$ in $(0, \lambda_0]$ is $\lambda_0(s)$. From the figures 1-2 can be seen in $[0, \infty)$ this function is continuous, convex down and is monotonely decreasing from λ_0 till. 0 .

We investigate $a(\Theta)$ in $[-1, 0)$. We have

$$a(-0) = +\infty, \quad a(-1) \leq a(1) = 0.$$

Consequently in $[-1, 0)$ /because of the continuity of $a(\Theta)$ / the equation $a(\Theta) = 0$ has at least one root. We assume

$$\lambda_1 = \sup \left\{ \Theta \in [-1, 0) : a(\Theta) = 0 \right\}.$$

As $a(\theta)$ is continuous function, so λ_1 is the greatest from such roots and in $(\lambda_1, 0)$ $a(\theta) > 0$. If in $[\lambda_1, 0)$ it were not monotone, then there would be such $\lambda_1 < \lambda_2 < \lambda_3 < 0$, that $a(\lambda_2) = a(\lambda_3) = 0$. Assuming in that case $s_0 = a(\lambda_2) > 0$ we would have (6) has at least three roots / λ_2 , λ_3 , $\lambda_0(s_0)$ / in the circle $|\theta| < 1$, which is not possible. So in $[\lambda_1, 0)$ $a(\theta)$ is monotonely increasing from 0 till $+\infty$. The function inverse to it in this interval is $\lambda_1(s)$. It can be easily seen $\lambda_1(s)$ is in $[0, \infty)$ continuous and monotonely increasing from λ_1 till 0 .

We assume for the chain ξ_t the states 0 and 1 are absorbing. In that case $b(\theta) = c(\theta) = 0$. Introducing for the function $\tilde{P}_\ell(s, \theta)$ in this case the notation $\tilde{P}^*_\ell(s, \theta)$ /see (5) / we get

$$\left[s - a(\theta)\right]\tilde{P}^*_\ell(s, \theta) = \theta^\ell - a(\theta)\tilde{P}^*_{\ell 0}(s) - \theta a(\theta)\tilde{P}^*_{\ell 1}(s)$$

and / $\theta \neq \lambda_j(s)$ /

$$s\tilde{P}^*_{\ell 0}(s) + s\lambda_0(s)\tilde{P}^*_{\ell 1}(s) = \lambda_0^\ell(s),$$

$$s\tilde{P}^*_{\ell 0}(s) + s\lambda_1(s)\tilde{P}^*_{\ell 1}(s) = \lambda_1^\ell(s), \tag{8}$$

$$s\tilde{P}^*_{\ell 1}(s)\left[\lambda_0(s) - \lambda_1(s)\right] = \lambda_0^\ell(s) - \lambda_1^\ell(s).$$

As in the last equality the left side is nonnegative, so at even ℓ $\lambda_0^\ell(s) \geq \lambda_1^\ell(s)$, which is equivalent to $\lambda_0(s) + \lambda_1(s) \geq 0$. If $\lambda_0 = 1$ and $a_1 = a_3 = \ldots = 0$, then $a(\theta)$ is even function in $[-1, 0) \cup (0, 1]$, from which $\lambda_1 = -1$ /fig.2/ follows. If $\lambda_0 < 1$, then $\lambda_0 + \lambda_1 \geq 0$ and for a certain $a_{2k-1} \neq 0$, then $a(-1) < a(1) = 0$, from which again follows $\lambda_1 > -1$. The theorem is proved.

3. Let us introduce the following notation /assuming

$\xi_0 = k > 1$ /

$$\tau_k = \inf\{t \geq 0 : \xi_t \leq 1\} \in [0, \infty),$$

and if $\tau_k < \infty$, then

$$\pi_k = \xi_{\tau_k + 0} \in \{0, 1\}$$

/if $\tau_k = u$, then $\xi_{u-0} > 1$, $\xi_{u+0} \leq 1$ and as the negative changes are equal to either -1 or -2 , so $\xi_{u-0} \in \in \{2, 3\}$ and $\xi_{u+0} \in \{0, 1\}$, and from $\xi_{u-0} = 3$ follows $\xi_{u+0} = 1$ /.

It can be easily seen /see the notations in the second part/

$$P\{\tau_k \in du, \pi_k = j\} = dP^*_{kj}(u), \quad k \geq 2, \quad 0 \leq j \leq 1,$$

and, consequently,

$$\varphi_{kj}(s) = M\left(e^{-s\tau_k}, \tau_k < \infty, \pi_k = j\right) = s\tilde{P}^*_{kj}(s).$$

According to (8)

$$\varphi_{k1}(s) = \frac{\lambda_0^k(s) - \lambda_1^k(s)}{\lambda_0(s) - \lambda_1(s)},$$

(9)
$$\varphi_{k0}(s) = -\lambda_0(s)\lambda_1(s)\frac{\lambda_0^{k-1}(s) - \lambda_1^{k-1}(s)}{\lambda_0(s) - \lambda_1(s)}.$$

As with probability 1 is $\tau_1 = 0$, $\pi_1 = 1$, so (9) is valid for any $k \geq 1$.

From (9) follows

$$P\{\tau_k<\infty\}=\varphi_{k1}(0)+\varphi_{k0}(0)=\frac{\lambda_0^k-\lambda_1^k-\lambda_1\lambda_0^k+\lambda_0\lambda_1^k}{\lambda_0-\lambda_1}=$$

$$=P\{\tau_{k-1}<\infty\}-(1-\lambda_0)(1-\lambda_1)\frac{\lambda_0^{k-1}-\lambda_1^{k-1}}{\lambda_0-\lambda_1}\,,$$

$$k\geq 2\,,$$

or

$$P\{\tau_k<\infty\}=1-(1-\lambda_0)(1-\lambda_1)\sum_{r=1}^{k-1}\frac{\lambda_0^r-\lambda_1^r}{\lambda_0-\lambda_1}\,.$$

So $P\{\tau_k<\infty\}=1$ then and only then, when $\lambda_0=1$. In the inverse case / $\lambda_0<1$ /

$$P\{\tau_k=\infty\}=(1-\lambda_0)(1-\lambda_1)\sum_{r=1}^{k-1}\frac{\lambda_0^r-\lambda_1^r}{\lambda_0-\lambda_1}>0.$$

4. We place in (5) $\theta=\lambda_j(s)$, $j=0,1$. We have

$$(10)\quad \left[s-c(\lambda_j(s))\right]\tilde{P}_{\ell 0}(s)+\left[s-b(\lambda_j(s))\right]\lambda_j(s)\tilde{P}_{\ell 1}(s)=\lambda_j^{\ell}(s),$$

$$j=0,1.$$

In our case

$$\begin{bmatrix}\tilde{P}_{00}(s) & \tilde{P}_{01}(s)\\ \tilde{P}_{10}(s) & \tilde{P}_{11}(s)\end{bmatrix}\begin{bmatrix}s-c(\lambda_0(s)) & s-c(\lambda_1(s))\\ (s-b(\lambda_0(s)))\lambda_0(s) & (s-b(\lambda_1(s)))\lambda_1(s)\end{bmatrix}=$$

$$=\begin{bmatrix}1 & 1\\ \lambda_0(s) & \lambda_1(s)\end{bmatrix}.$$

As $\lambda_1(s)-\lambda_0(s)<0$, so

$$(11)\quad D(s)=\begin{vmatrix} s-c(\lambda_0(s)) & s-c(\lambda_1(s)) \\ (s-b(\lambda_0(s)))\lambda_0(s) & (s-b(\lambda_1(s)))\lambda_1(s) \end{vmatrix}\neq 0,$$

$$s>0.$$

(10) is a system of linear equations with determinant $D(s)\neq 0$.

Consequently we have proved the next /see (5) /

Theorem 2.

$$(12)\quad \tilde{P}_l(s,\theta)=\frac{\theta^l-[a(\theta)-c(\theta)]\tilde{P}_{l0}(s)-[a(\theta)-b(\theta)]\theta\tilde{P}_{l1}(s)}{s-a(\theta)},$$

$$(0<\theta<\lambda_0\leq 1),$$

where $\tilde{P}_{l0}(s)$ and $\tilde{P}_{l1}(s)$ are determined by (10) .

5. We find necessary and sufficient conditions for the chain ξ_t , $t\geq 0$ to be ergodic. It is clear, that all the states 0 , 1 , 2 ,... must be communicating. For this purpose it is enough to assume the class of states $\{0,1\}$ is not closed, $a_1>0$ and in the sequence $\{a_k\}$ there is an infinite number of positive terms. Furthermore, if the chain is ergodic, then with probability 1 we can get from any state $k\geq 2$ to the class $\{0,1\}$, which is equivalent to /see (3) /

$$(13)\qquad P\{\tau_k<\infty\}=1 \sim \lambda_0=1 \sim a'(1)\leq 0.$$

In the chain with communicating states the limits

$$\lim_{t\to\infty} P_{lk}(t)=P_k\geq 0$$

exist and are independent of l . Multiplying the both sides of (10) by s , we get /as $s\to 0$ and taking

into account (13) /

$$c(\lambda_1)P_0+\lambda_1 b(\lambda_1)P_1=0. \tag{14}$$

The coefficients of this equation cannot be equal to zero at the same time. Really, if $\lambda_1>-1$, then $c(\lambda_1)\le c(-\lambda_1)<c(1)=0$. If $\lambda_1=-1$ then $a(\theta)$ is even function and $c(-1)=b(-1)=0$ can be only at even $b(\theta)$ and $c(\theta)$ which is impossible, as in such case the classes $\{0,2,\ldots\}$ and $\{1,3,\ldots\}$ would be closed. E.g. let $b(\lambda_1)=0$, then $c(\lambda_1)\neq 0$ and from (14) follows $P_0=0$. This fact contradicts the ergodicity of ξ_t, and so the fact that all the states $\{0,1,2,\ldots\}$ are communicating. Indeed, from Rouché's theorem follows in $(-1,0)$ $b(\theta)\neq 0$ /the only root of equation $b(\theta)=0$ in the circle $|\theta|<1$ is in the interval $(0,1)$ if $b'(1)>0$; in the inverse case $b(\theta)>0$ in $(0,1)$ and so $b(\theta)<0$ in $(-1,0)$ /. So from $b(\lambda_1)=0$ follows $\lambda_1=-1$ and $a(\theta)$, $b(\theta)$ are even functions, which is equivalent to the fact, that the class $\{1,3,\ldots\}$ is closed. So

$$c(\lambda_1)<0,\ \lambda_1 b(\lambda_1)>0,\ P_0=c_*\lambda_1 b(\lambda_1),\ P_1=-c_* c(\lambda_1), \tag{15}$$

where $c_*\ge 0$ is an unknown constant.
Multiplying the both sides of (12) by s, we get as $s\to 0$

$$P(\theta)=\sum_{k=0}^{\infty}P_k\theta^k=\left[1-\frac{c(\theta)}{a(\theta)}\right]P_0+\left[1-\frac{b(\theta)}{a(\theta)}\right]\theta P_1=$$

$$=c_*\left[\lambda_1 b(\lambda_1)\left(1-\frac{c(\theta)}{a(\theta)}\right)-c(\lambda_1)\left(1-\frac{b(\theta)}{a(\theta)}\right)\right], \tag{16}$$

$$0<\theta<1.$$

Finding the limit as $\theta\to 1$ we have

$$(17)\qquad 0 \leq c_* \left[\lambda_1 b(\lambda_1) \left(1 - \frac{c'(1)}{a'(1)} \right) - c(\lambda_1) \left(1 - \frac{b'(1)}{a'(1)} \right) \right] \leq 1.$$

Consequently $c_* \in (0, \infty)$ then and only then if the expression in the brackets in (17) is bounded. In such case the chain is ergodic and the right inequality in (17) will be an equality.

So at last we get

Theorem 3. The chain ξ_t, $t \geq 0$ is ergodic then and only then if 1. all the states are communicating, 2. $a'(1) < 0$, 3. $c'(1) < \infty$ and $b'(1) < \infty$. The generating function of the ergodic distribution is of the next form

$$(18)\qquad P(\theta) = \frac{\lambda_1 b(\lambda_1) \left[1 - \frac{c(\theta)}{a(\theta)} \right] - c(\lambda_1) \theta \left[1 - \frac{b(\theta)}{a(\theta)} \right]}{\lambda_1 b(\lambda_1) \left[1 - \frac{c'(1)}{a'(1)} \right] - c(\lambda_1) \left[1 - \frac{b'(1)}{a'(1)} \right]},$$

where λ_1 is the greatest root of the equation $a(\theta) = 0$ in $[-1, 0)$.

Similar problem was solved by us for time-dependent case, too. On the same way we get the direct system of Kolmogorov differential equations and it leads to a linear differential equation for the generating function of transition probabilities. Solving this equation we get an expression for the generating function which includes the first two probabilities $P_{i0}(t, s)$ and $P_{i1}(t, s)$ from the desired distribution. These probabilities may be got from the first two equations as the the solution of a system of Volterra integral equations.

The authors thank Professor I.I. Ezhov /Kiev/ for the attention to the present work.

References

[1] Хинчин А.Н.: Работы по математической теории массового обслуживания, физматгиз, Москва, 1963.

[2] Гнеденко Б.В.,Коваленко И.Н.: Введение в теорию массового обслуживания, изд.Наука, Москва, 1966.

[3] Єлов І.І.: Про одну задачу Такача, Вісник Київського університету, сер. матем. і механ.,№5, вип. 2, 1962, стр.161-163.

[4] Корнійчук М.Т.,Маркова Л.Н.: Исследование одного класса систем массового обслуживания с дискретным временем, препринт Института математики АН УССР, Киев, 1977/9.

Mathematical Institute of the UkrSSR Academy of Sciences
252601 KIEV-4, GSP, ul.Repina 3.
Soviet Union

ABOUT THE BLOCKING EFFECT IN TELEPROCESSING NETWORKS

H. LÖFFLER

1. Introduction

The aim of the following explanation consists in the application of blocking queuing systems to performance and behaviour evaluation of teleprocessing networks and of access systems of computer networks, resp.

Let us consider a teleprocessing network with batch processing (fig. 1). The jobs are introduced into the network from terminals T and sended through transmission channels to a so-called terminal-interface-processor TIP. This is a hardware and software component (front-end processor) in the network which realizes [1] :

- the interface between the decentral components and the processing unit (CS)
- multiplex connection of the terminals
- the standard teleaccess work.

In the central computer system CS takes place the processing of the user jobs. Between the TIP and the CS is connected a channel/channel adaptor. This is necessary because the hardware and the operating systems of the TIP-computer and the central computer system differ. The presented configuration is similar to the access component of a computer network in the German Democratic Republic (CS stands for the core of the computer network).

Some of the interesting performance characteristics of the access systems are:

Rate of throughput D_R: Mean number of demands handled in a certain time T.

Throughput D: It is the "work" of the system in a time T, $D = D_R T$.

Maximum throughput $D_{max} = \max (D_R \mid \forall i: \eta_i < 1)$
It is the maximum throughput possible under steady-state conditions. η_i is the relative utilization of the i-th component.

Mean delay time T_v : (Weighted) sum of all waiting and service times

Normalized delay time T_{vN}: Mean delay time related to teh sum of all service times

Quality factor $Q = \sum_i \eta_i / T_{vN}$ This measures takes in consideration that a higher throughput is accompanied of increasing delay times.

It is the aim of the following to value the performance of the access system and to determine the bottlenecks between the user and the central processing unit. An approximate performance estimation is possible by application of the theory of markovian queuing systems with infinite queues (Jackson's theorem). A better approximation seems to be the application of the theory of queuing systems with finite queues.

2. Tandem blocking systems

Let us consider a sequence of two queues. The first station may have an infinite queue size ($s_1 = \infty$), the second queue volume may be limited ($s_2 < \infty$). Then the second station blocks the first one if it contains s_2+1 demands (s_2 demands are waiting, 1 demand is served). Such a sequence of queuing stations we call tandem blocking system BL (∞, s_2). Investigations of this typ of coupled queuing systems and of the typ BL ($\infty, s_2, \ldots, s_n$) are published for example in [2,3,4,5]. We extend this investigations to the case of parallel stations in the first and in the second system.

For the present we dwell on the system BL (∞, s_2) with the characteristics as follows:

- exponentially distributed inter-arrival times of the demands enternig the system,
- service times exponentially distributed and independent
- service discipline: FIFO
- service in the first server can take place only if the second buffer not is totaly filled (thad meens blocking before service)

For the system BL(∞, s_2) with the characteristics mentioned above we can derive from the state graph the following system of steady-state probabilities (P(i,j):(probability for i demands in the first station and j demands in the second)

$$
\begin{aligned}
\lambda P(0,0) &= P(0,1)\mu_2 \\
P(i,0)(\lambda+\mu_1) &= P(i-1,0)\lambda + P(i,1)\mu_2;\ i>0 \\
P(0,j)(\lambda+\mu_2) &= P(1,j-1)\mu_1 + P(0,j+1)\mu_2 \\
&\qquad 0<j<s_2+1 \\
P(o,s_2+1)(\lambda+\mu_2) &= P(1,s_2)\mu_1 \qquad (1) \\
P(i,s_2+1)(\lambda+\mu_2) &= P(i-1,s_2+1)\lambda + P(i+1,s_2)\mu_1 \\
&\qquad i>0 \\
P(i,j)(\lambda+\mu_1+\mu_2) &= P(i-1,j)\lambda + P(i+1,\ j-1)\mu_1 \\
&\quad + P(i,j+1)\mu_2 \\
&\quad (0<j<s_2+1;\ i>0
\end{aligned}
$$

This is a system of infinite number of equations, which not can be solved exactly. For an approximate solution has been proposed different approdches (see e.g. [4,5,6]).

We has used for an approximate solution two approaches:

(1) assumption, that a demand departes from the first system only if in the second system are less than s_2+1 demands

(2) assumption, that a demand which cannot enter the second system is handled again from the first one

With this assumptions the blocking queuing system $BL(\infty, s_2)$ can be calculated approximately. The blocking of the first part of the tandem queuing system by the second (no free buffer capacity) provoces that the input into the second is

$$\lambda^* = \lambda/(1 - P_{BL})$$

P_{BL} is the blocking probability. The mean service rate in the first system is

$$\mu_1^* = \mu_1(1 - P_{BL})$$

For calculating of the system's parameter it is necessary to know the probabilities

$$P(i,) = \sum_{j=0}^{s_2+1} P(i,j)$$

(Probability i demands in the first system and an arbitrary number of demands in the second queuing system) and

$$P(,j) = \sum_{i=0}^{\infty} P(i,j)$$

(probability for an arbitrary number of demands in the first queuing system and j demands in the second).
From the equations (1) we obtain for the macro-steady-state probabilities

$$P(i,) = (\varrho_1^*)^i \, P(0,); \; i > 0$$
$$P(0,) = 1 - \varrho_1^* \quad \text{if}$$
$$\varrho_1^* = \lambda/[\mu_1(1 - P(,s_2+1))] < 1,$$

and

$$P(,j) = (k^*)^j \, P(,0); \quad 0 < j \leq s_2+1 \qquad (2)$$
$$P(,0) = (1 - k^*)/[1 - (k^*)^{s2+2}]$$
$$P(,s_2+1) = (k^*)^{s_2+1} \, P(,0)$$

with

$$k^* = \mu_1(1 - P(0,))/\mu_2 \neq 1$$

and

$$P(\ ,0) = (s_2+2)^{-1} \cdot P(\ ,s_2+1)\ ;\quad k^* = 1$$

The equations (2) hold for the steady-state conditions

$$\varrho_1^* = \varrho_1/(1-P(\ ,s_2+1)) < 1$$

$$\varrho_2 = \varrho_1 k = \lambda/\mu_2 < 1\ ;\ k = \mu_1/\mu_2 \qquad (3)$$

From (2) follows: the tandem queuing system $BL(\infty,s_2)$ can be calculated by decomposition into two queuing systems. It must be known:

- for the first partial queuing system: the probability that in the second partial queuing are s_2+1 demands (s_2 demands waiting and one served). This is the blocking probability:

  $$P_{BL} = P(\ ,s_2+1).$$

  Caused by the blocking phenomenon the first partial queuing system works with the virtual intensity of service $\mu_1^* = \mu_1(1-P_{BL})$.
- for the second partial queuing system: the probability that the first partial queuing system is empty:

  $$P(0,\ ) = P_{EMPTY}$$

The approximate calculation holds for the following assumptions of independancy:

- $P_{EMPTY} = P(0,\ )$ not depends on the number of demands in the second queuing system;
- $P_{BL} = P(s_2+1$ demand in the 2^{nd} system) is independent of the number of demands in the first system.

Fig. 6 shows some results. If $k = \mu_1/\mu_2 < 1$ we obtain a sufficient conformity of analytical results and simulation. If $k > 1$ and $\varrho_2 > 0{,}5$ the analytical results are lower than the results obtained by simulation. This means, that the assumptions of independancy only holds for blocking systems with low values of blocking probability. A better conformity of analytical results

and simulation for $k > 1$ can be obtained if we put

$$P^{*}_{BL} = q\, P_{BL};\quad q = \varrho_2/\varrho_1 \qquad (4)$$

An approximate approach for queuing systems in series of the class BL $(\infty, s_2, \ldots s_n)$ is the following:
(1) decomposition of the total system in subsystems of the class BL(∞, s_i);
(2) the calculation starts at the n-th system (see [4,5,6]).

In order to calculate the performance of teleprocessing networks by means of the theory of blocking queuing systems, it is necessary to consider structures of the following classes:
- Parallel queuing systems of infinite queue capacity which are connected with a queuing system of finite queue capacity (fig. 4) we call such a system tandem-blocking system typ union.
- One queuing system of infinite buffer capacity is connected with 2 parallel systems of finite queue capacities (so-called tandem-blocking system typ branching, see fig. 5).

Based on the steady-state graph of the systems it is possible to derive a system of equations (FIFO discipline, exponentially distributed inter-arrivel times and service times).
For the system typ union we obtain:

$$P(0,\ ,\ ) = 1 - \varrho^{*}_{11}$$

$$P(i,\ ,) = (\varrho^{*}_{11})^{i}\, P(0,\ ,\ );\quad \varrho^{*}_{11} = \frac{\lambda_{11}}{\mu_{11}(1-P(\ ,\ ,s_2+1))}$$

$$P(\ ,0,\ ) = 1 - \varrho^{*}_{12}$$

$$P(\ ,j,\ ) = (\varrho^{*}_{12})^{j}\, P(\ ,0,\ );\ \varrho^{*}_{12} = \frac{\lambda_{12}}{\mu_{12}(1-P(\ ,\ ,s_2+1))} \qquad (5)$$

$$P(\ ,\ ,0) = (1 - u)/(1 - u^{s_2+2})$$

$$P(\ ,\ ,k) = u^{k}\, P(\ ,\ ,0)$$

$$u = \frac{\mu_{11}(1 - P(0,\ ,\ )) + \mu_{12}(1 - P(\ ,0,\ ))}{\mu_2}$$

In (4) means P(i,j,k) the probability of i demands in the queuing system 11, j demands in the queuing system 12 and k demands in the queuing system 2.
From (4) follows

- the first systems (in parallel) can be calculated if we know the blocking probability $P_{BL} = P(, ,s_2+1)$;
- the second system can be considered as a $M/M/1/s_2$ - system if are known the probabilities that no demands exists in the systems with infinite queue capacities;
- the blocking probability and the probabilities for the empty parallel queuing systems are available from the first and last equation in (4) by iteration;
- the results can be extended to an arbitrary number of parallel queuing systems connected in series with a queuing system of finite queue capacity;
- the tandem-blocking system BL (∞,s_2) is a special case of the structure considered here.

Fig. 7 illustrates the normalized delay time of a tandem blocking system typ union versus the relative input-rate into the system;

$\varrho_{11} = \varrho_{12}$

In similar manner has been derived relations for the tandem-blocking system typ branching by Oehme [7].

3. Application of the theory

We use the theory of tandem-blocking systems presented above to the study of the model of a teleprocessing network (fig. 1).
The study begins with the analisis of the flow of demands (fig.2).
We assume:

- all demands at the terminals are Poisson-distributed,
- equal intensities of inputs and outputs,
- service time of the TIP and of the CS exponentially distributed;
- exponentially distributed lengthes of the user demands in the transmission channels and in the channel/channel adapter,
- service disciplinies: FIFO.

For a batch-processing teleprocessing system important is the rate of throughput D_R. The rate of throughput must be everytime lower than the value of D_{max} (maximum rate of throughput). Thus in our model must hold for all service facilities:

$$\forall_j : \lambda_j < \mu_j \, (1 - P_{BL,j+1}) = \mu_j^*$$

Another important measure is the quality factor of the system components terminal ... CCA ... terminal . The quality factor illustrates the interaction between the utilization of the system and the normalized delay times. A remarkable results is drawn in fig. 8. It demonstrates the influence of the storage capacity of the front-end-processor TIP to the behaviour of the total teleprocessing system. The curves holds ofr lengthes of messages of 1000 bytes, capacities of the channels of 2 bit/s , capacity of the CCA of 5 bit/s , servic time of the TIP of 10 units of time. From fig. 8 we see, that a small storage capacity of the TIP limits the performance of the total system. The blocking effect in this case already appears at small throughput rates.

Bibliography

[1] Irmscher, K. and Löffler H.: Design and performance evaluation of control programs for terminal access to computer networks. TU-Informationen 08-08-79

[2] Labetoulle, J. and G. Pujolle: A study of queueing network with deterministic service and application to computer networks. Acta Informatica 7 (1976), 183 - 195

[3] Hillier, F.S. and R.W. Boiling: Finite queues in series with exponential or Erlang service times - a numerical approach. Op. Res. (1966) 286 - 303

[4] Krämer, W.: Untersuchung von Systemen mit seriellem Warten. Diss. Stuttgart 1974

[5] Löffler, H. und P. Tschernokoschew: Das Zweiphasenbedienungssystem mit Blockierung BS (∞,s). Wiss.Z. Technische Universität Dresden, 26(1977), Nr. 2, 349 - 353

[6] Löffler, H.: Eigenschaften von Tandem-Bedienungssystemen mit Blockierung. msr 20(1977) 617 - 619

[7] Oehme, W.: work to be published 1980

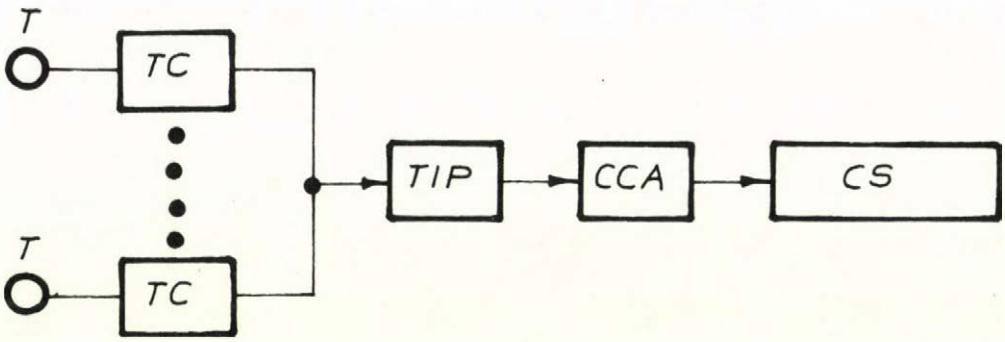


fig. 1 : Configuration of a specific teleprocessing network

(T : terminal ; TC : transmission channel ;
CCA : channel-to-channel adaptor ;
CS : computer system)

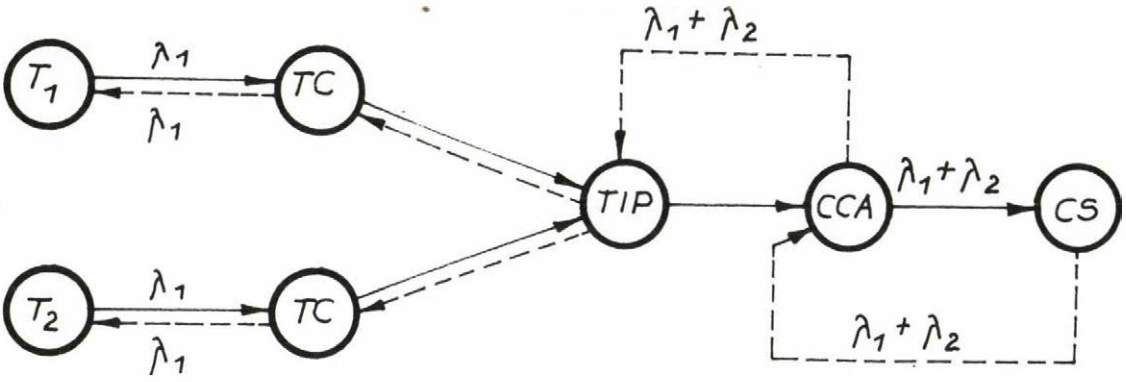
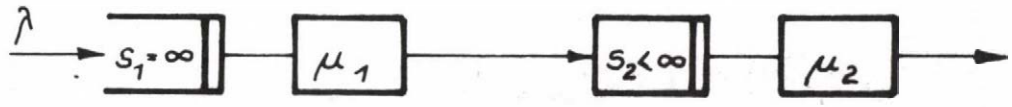
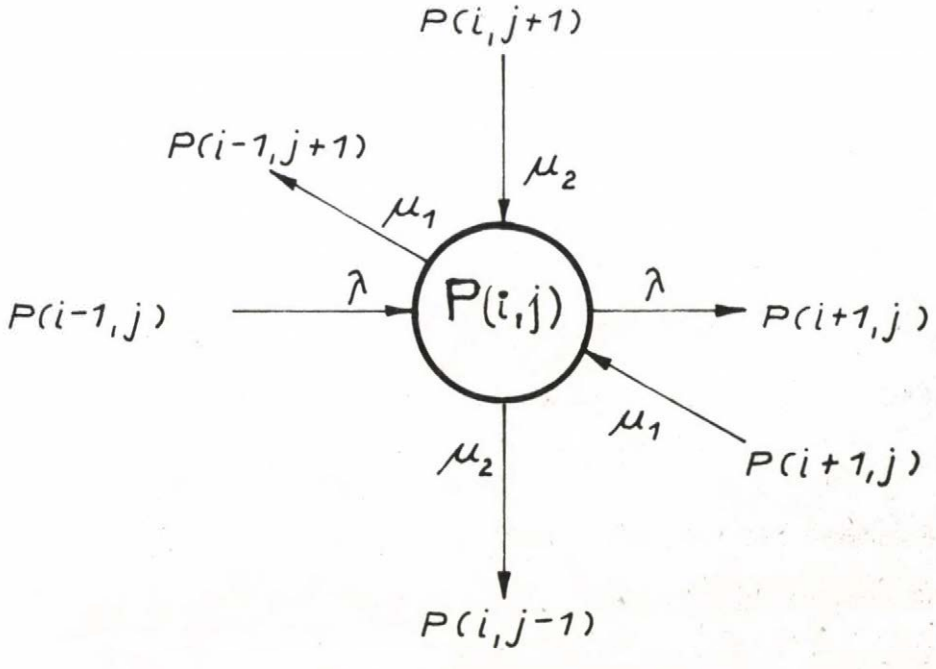


fig. 2 : Flow diagramm (only 2 terminals)



a)



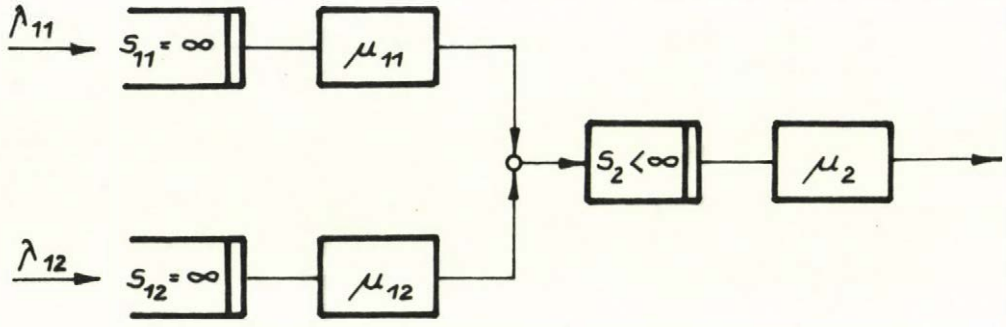
b)

$0 < j \leq S_2 + 1$

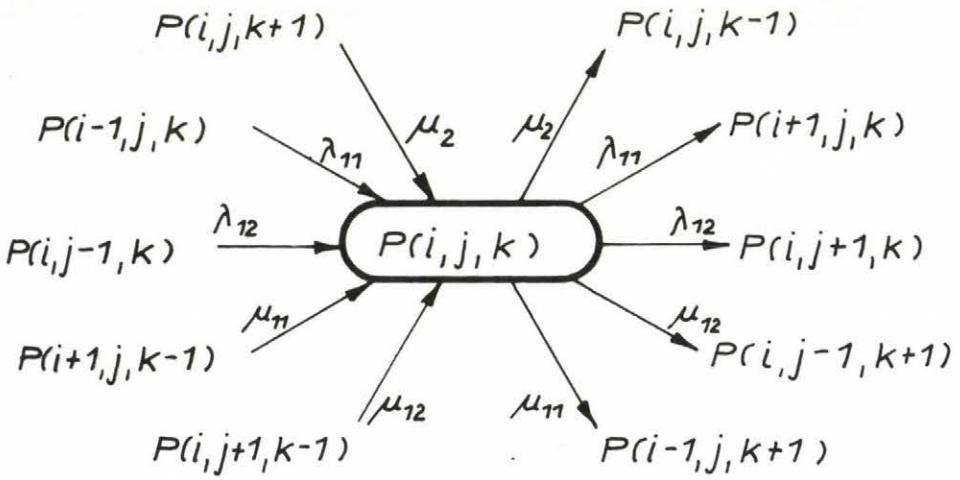
$i > 0$

fig. 3 : a) Tandem-blocking system BL (∞, S_2)

b) Part of the state graph



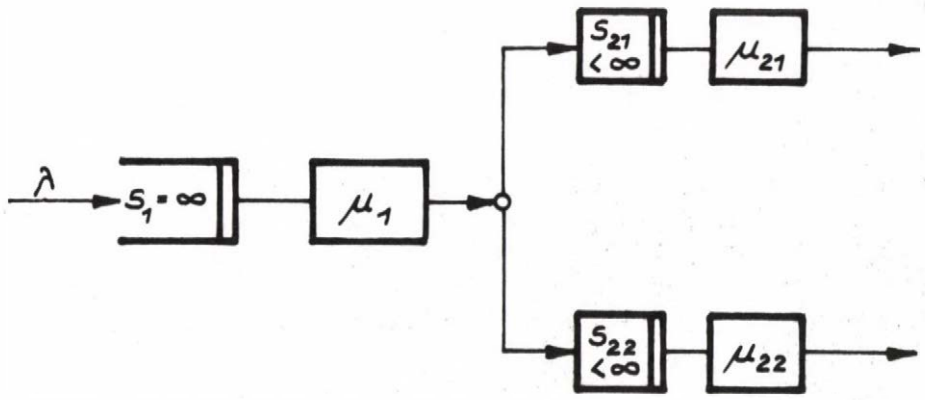
a)



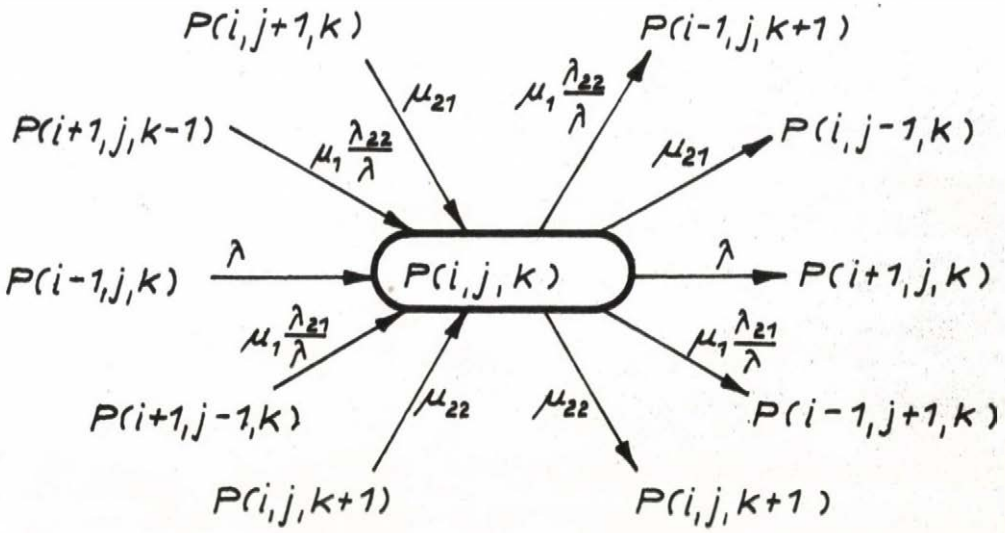
b) $i, j > 0 \; ; \; 0 < k < S_2 + 1$

fig. 4: a) Tandem - blocking system typ „union"

b) part of the state graph



a)



b)

$$i > 0 \;,\; 0 < j < S_{21}+1 \;;\; 0 < k < S_{22}+1$$

fig.5: a) Tandem - blocking system typ „branching"

b) part of state graph

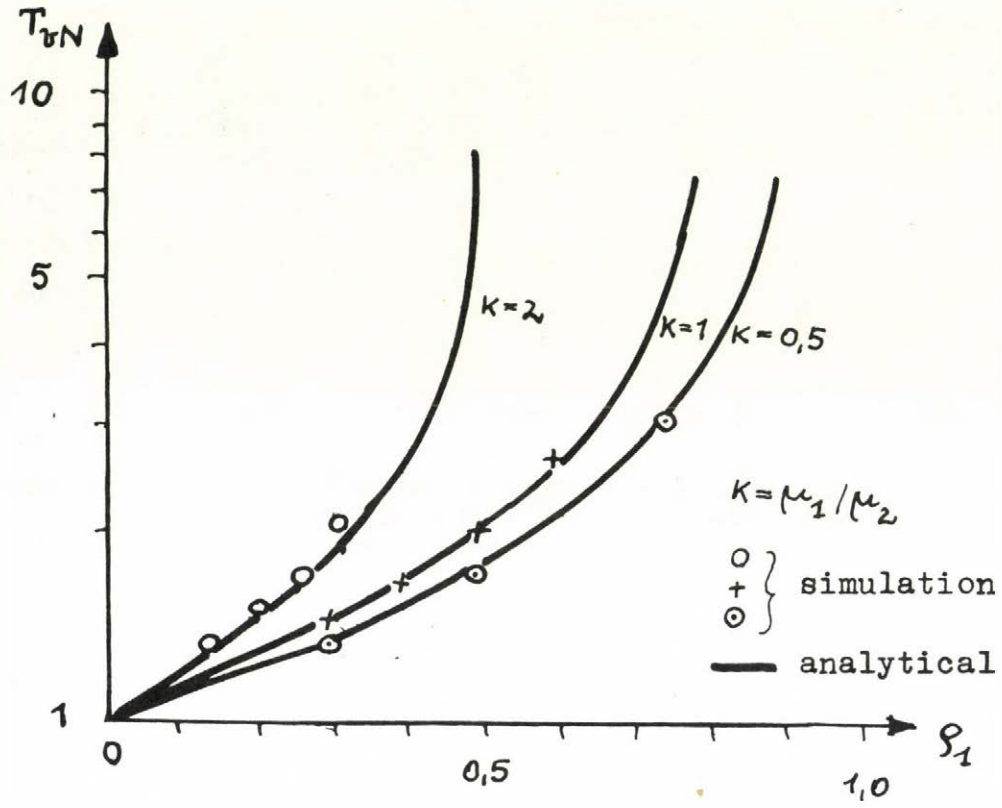


fig. 6: Normalized delay time T_{vN} versus $\varrho_1 = \frac{\lambda}{\mu_1}$
of a tandem queuing system BL(∞,4);
simulation and analytical results

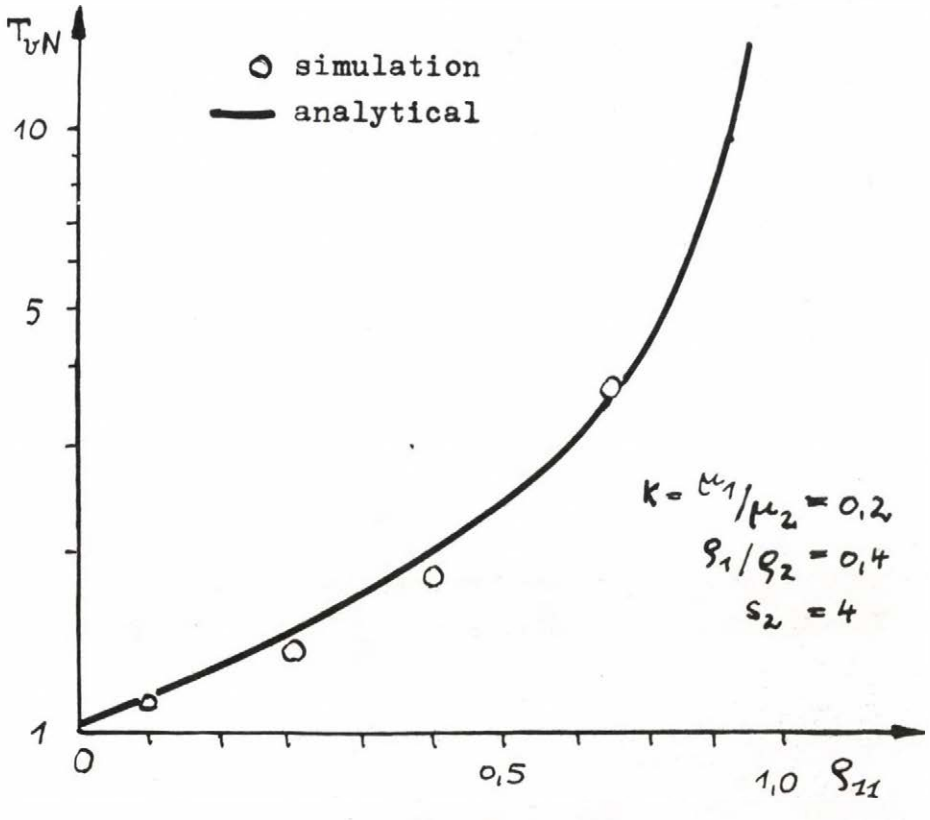


fig. 7: Normalized delay time of a tandem-blocking system typ union

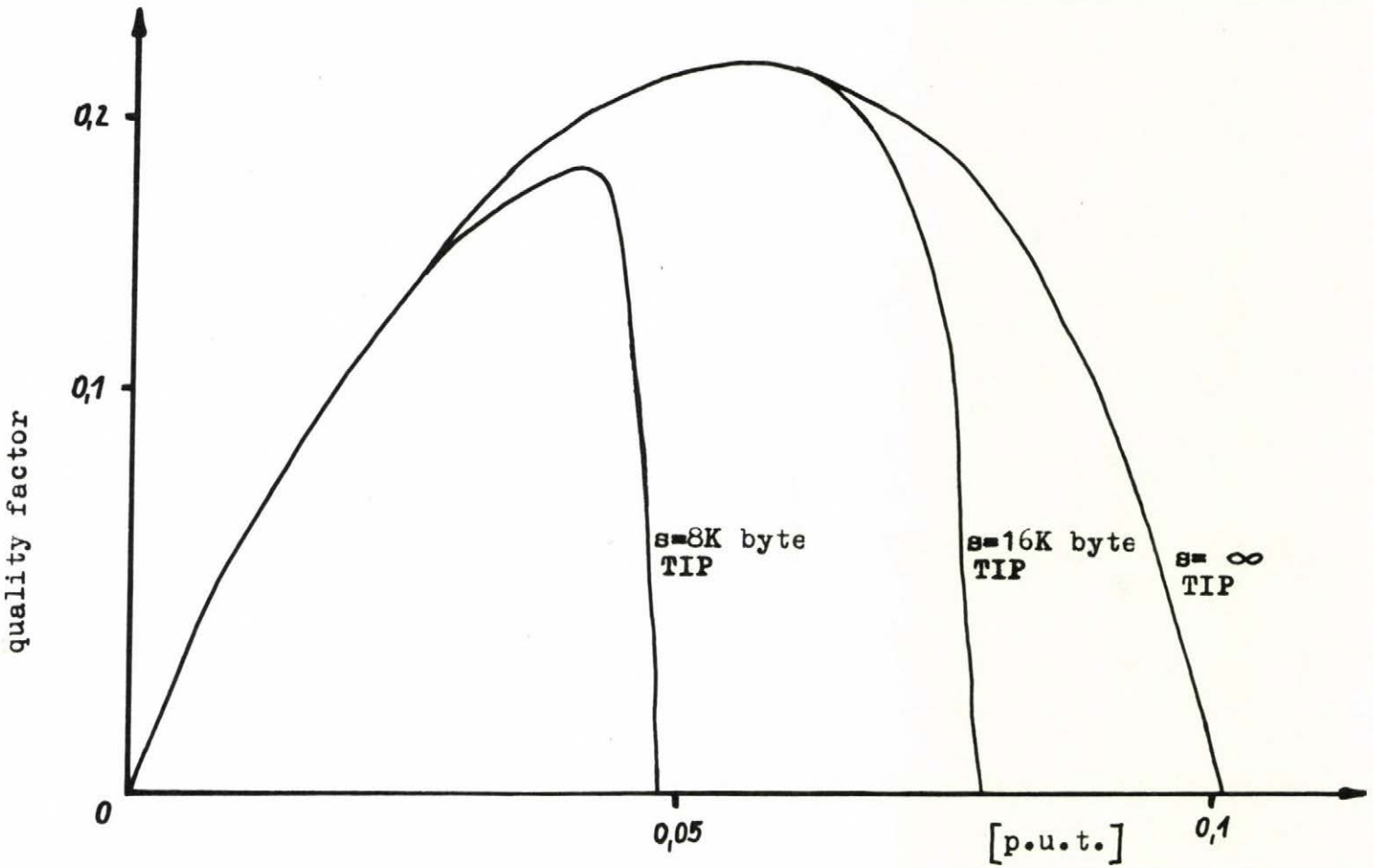


fig. 8: Quality factor versus throughput - rate

On the asymptotic network delay in a model of packet switching

by

Tamás F. Móri

Department of Probability Theory

Loránd Eötvös University, Budapest

1. Introduction

Consider a network transmission path involving $n+1$ nodes (numbered from 0 to n) and n transmission lines. The channels are assumed to be noiseless, perfectly reliable and to have the same (say unit) capacity. Our task is to route a message of length k through the network. We assume that this transmission path is a part of a larger network, hence some other traffic can interfere with the transmission, i.e. our transmission lines are not sure to be free. This fact will be taken into consideration by assuming that within a given unit period each channel has the same probability p of being free, independently of each other and of their state in the past.

We can proceed in two different ways according to the message and packet switching method of transmission (see [1]). The message can be transmitted step by step in its entirety or it can be decomposed into k packets of unit length, which are forwarded through the network simultaneously. We are interested in the network delay for both switching methods.

Denote T'_{kn} resp. T_{kn} the network delay for message and packet switching. We are going to analyse the asymptotic behaviour of these quantities as n tends to infinity.

2. Results for message switching

Denote X_j the total time that the message spends passing through the jth channel i.e. the sum of the waiting time in queue

at the $(j-1)$th node and the message transmission time. The latter is proportional to the length of the message. Here we chose the channel capacities to be unit, therefore the transmission time amounts to k. The queueing time is the waiting time for the first "free" state of the channel, it takes the value i with probability $p(1-p)^i \qquad i=0,1, \ldots$

$X_1, X_2, \ldots$ are independent identically distributed random variables with mean $\mu_k=k-1+p^{-1}$ and variance $\sigma_k^2=p^{-2}-p^{-1}$, hence the asymptotic distribution of the network delay is obtained from classical results of probability theory.

PROPOSITION 1

a) $\lim_{n\to\infty} n^{-1}T'_{kn}=\mu_k$ *a.s. and in* L_2 *too.*

b) $n^{1/2}\sigma_k^{-1}(n^{-1}T'_{kn}-\mu_k) \to_d N(0,1)$

where $\to_d$ *stands for convergence in distribution and* $N(0,1)$ *denotes the standard normal distribution.* □

Here we assume that once a channel becomes free, our message reserves it for all the transmission time. Actually it would be reasonable to suppose the waiting time before transmissing a long message to be longer. E.g. if the message should wait for k consecutive "free" states of the channel, then

$$\mu_k=\sum_{i=1}^{k} p^{-i} \qquad \text{and} \qquad \sigma_k^2=p^{-k}(1-p)\sum_{i=1}^{k} p^i\mu_i^2 \ .$$

3. Packet switching, a reformulation of the problem

If numbered packets are sent successively through the network then they move at different speed rates: the one having the greater number needs the longer time for travelling through the network, since it often has to wait for the sake of packets with smaller

number. Is this additional queueing significant compared with the transmission time of the number one packet? We shall see it isn't.

Denote X_{ij} the time that the ith packet spends at the jth channel after the preceding packet has passed through. Then

$$X_{ij} \qquad i=1,2,\ \dots\ ,k; \quad j=1,2,\ \dots\ ,n$$

are independent random variables with negative binomial distribution of first order, further

$$\begin{aligned} &T_{1,j}=X_{1,1}+X_{1,2}+\ \dots\ +X_{1,j} \\ &T_{ij}=(T_{i,j-1}\vee T_{i-1,j})+X_{ij} \qquad T_{i,0}=0 \qquad i\geq 2 \end{aligned} \tag{1}$$

Let us form the matrix $(X_{ij})_{i=1,\ \dots\ ,k;\ j=1,\ \dots\ ,n}$ and consider all the paths connecting $X_{1,1}$ with X_{kn}, which step from any element to its right or lower neighbour in the matrix. These $\binom{n+k}{n}$ paths are said to be admissible. For each admissible path let us form the sum of the elements reached along the way. By induction it can easily be seen that the maximum of these sums equals T_{kn}. This reformulation provides an easy approach to the problem.

4. Limit theorems

Since the knowledge of the exact law according to which the random variables X_{ij} distribute provides no additional information on the asymptotic behaviour of T_{kn}, therefore throughout the rest of this paper $(X_{ij})_{i=1,\ \dots\ ,k;\ j=1,\ \dots\ ,n}$ will denote an arbitrary array of independent identically distributed random variables with mean μ and variance σ^2. The dependence of T_{kn} on the distribution of X_{ij} will be expressed, if necessary, by writing $T_{kn}(X_{ij})$. Note that in our model of packet switching $\mu=p^{-1}$ and $\sigma^2=p^{-2}-p^{-1}$.

PROPOSITION 2

$$\lim_{n\to\infty} n^{-1}T_{kn} = \mu \qquad a.s.\ and\ in\ L_2.$$

Proof. Suppose first that $X_{ij} \geq 0$. Let m be a fixed positive integer and $(d-1)m < n \leq dm$. An upper estimate of $T_{k,dm}(X_{ij})$ can be obtained in the following way:

Let us divide the matrix into $k \times m$ cells by holding up the elements of the same row by m's, and for each admissible path let us form the sum of all the elements standing in the cells reached on the way. Thus

$$T_{k,dm}(X_{ij}) \leq T_{k,m}(S_{d,i,j})$$

where

$$S_{d,i,j} = \sum_{r=1}^{n} X_{i,(j-1)d+r} .$$

From this we have

$$n^{-1} T_{kn}(X_{ij}) \leq ((d-1)m)^{-1} T_{k,dm}(X_{ij}) \leq m^{-1} T_{km}((d-1)^{-1} S_{d,i,j})$$

Applying the strong law of large numbers we obtain that

$$\limsup_{n \to \infty} n^{-1} T_{kn}(X_{ij}) \leq \frac{k+m-1}{m} EX_{ij} \qquad \text{a.s.}$$

Since m can be arbitrary large, therefore

$$\limsup_{n \to \infty} n^{-1} T_{kn}(X_{ij}) \leq \mu \qquad \text{a.s.} \qquad (2)$$

This result can easily be extended to the general case by applying (2) to the truncated variables $\hat{X}_{ij}(a) = (-a) \vee X_{ij} \wedge a$ $(a>0)$.

Finally, since

$$\sum_{j=1}^{n} X_{1,j} + \sum_{i=2}^{k} X_{i,n} \leq T_{kn}(X_{ij})$$

therefore

$$\mu \leq \liminf_{n \to \infty} n^{-1} T_{kn}(X_{ij}) ,$$

and the proof of the a.s. convergence is completed.

The convergence in L_2 follows from the fact that $|n^{-1} T_{kn}|$ is majorized by the L_2-convergent sequence $n^{-1} \sum_{j=1}^{n} \sum_{i=1}^{k} |X_{ij}|$. □

The above proposition asserts that the average speeds of the

different packets do not differ asymptotically. It means also that in our model the network delay is much more less for packet switching than for message switching.

PROPOSITION 3

$$n^{1/2}\sigma^{-1}(n^{-1}T_{kn}-\mu) \;\to_d\; \tau_k(W_1,W_2,\ \dots\ ,W_k)$$

where W_1, W_2, ... , W_k *are independent standard Wiener processes and* τ_k *denotes the following functional operating on functions* $f=(f_1,\ \dots\ ,f_k)\colon [0,1] \to R^k$

$$\tau_k(f_1,\ \dots\ ,f_k)= \sup\{ \sum_{i=1}^{k} (f(t_i)-f(t_{i-1})) \colon 0=t_o\leq t_1\leq \dots \leq t_{k-1}\leq t_k=1 \}\ .$$

Proof. The proof is an immediate application of the multidimensional invariance principle (see [2]).

For $i=1,2,\ \dots\ ,k$ let $S_{in}(t)$ $(0\leq t\leq 1)$ denote a random broken line with breaking points

$$\Big(r/n,\ \ n^{-1/2}\sigma^{-1} \sum_{j=1}^{r} (X_{ij}-\mu) \Big) \qquad r=0,1,\ \dots\ ,n \quad .$$

Then

$$(S_{1,n},S_{2,n},\ \dots\ ,S_{k,n}) \to_d (W_1,W_2,\ \dots\ ,W_k)$$

in $C_{R^k}[0,1]$.

One can readily verify that

$$|\tau_k(S_{1,n},\ \dots\ ,S_{k,n})- n^{1/2}\sigma^{-1}(n^{-1}T_{kn}-\mu)| \to 0 \qquad \text{a.s.}$$

Since τ_k is continuous on the space $C_{R^k}[0,1]$, the proof is completed. □

It appears to be very difficult to determine the exact distribution of $\tau_k(W_1,\ \dots\ ,W_k)$ in general. In the simplest non-trivial case, i.e. when $k=2$, the calculation can be carried out as follows:

Denote $\hat{W}_1=(W_1+W_2)/\sqrt{2}$ and $\hat{W}_2=(W_1-W_2)/\sqrt{2}$. Then $\hat{W}_1$ and $\hat{W}_2$ are independent standard Wiener processes and $\tau_2(W_1,W_2)$ can be written in the form

$$\tau_2(W_1,W_2)= \sqrt{2}\ \sup_{0\leq t\leq 1} \hat{W}_2(t)+(\hat{W}_1(1)-\hat{W}_2(1))/\sqrt{2}$$

The joint distribution of the terms ond the right-hand side is well-known, from this a straightforward calculation yields the joint density function of $(W_1(1),\ \tau_2(W_1,W_2))$:

$$h_{1,2}(x,y)=\varphi(x)\varphi(y)+\Phi(x)y\Phi(y) \quad \text{if } x<y \text{ and } 0 \text{ otherwise}$$

where φ and Φ stand for the standard normal distribution and density function. The corresponding distribution function is

$$H_{1,2}(x,y)= \begin{cases} \Phi(x)\Phi(y)-x\Phi(x)\varphi(y)-\varphi(x)\varphi(y) & \text{if } x<y \\ \Phi^2(y)-y\varphi(y)\Phi(y)-\varphi^2(y) & \text{if } x\geq y \end{cases}$$

The expectation and the covariance matrix of this distribution are

$$(0,\ 2/\sqrt{\pi}) \quad \text{and} \quad \begin{pmatrix} 1 & 1/2 \\ 1/2 & 2-4/\pi \end{pmatrix} .$$

PROPOSITION 4

$$\limsup_{n\to\infty} n^{1/2}(2\sigma^2 \log\log n)^{-1/2}(n^{-1}T_{kn}-\mu) = \sqrt{k}$$

and

$$\liminf_{n\to\infty} n^{1/2}(2\sigma^2 \log\log n)^{-1/2}(n^{-1}T_{kn}-\mu) = -1$$

with probability 1.

Proof. As in the proof of Proposition 3, we use again invariance principles: the functional version due to Strassen of the law of iterated logarithm (see [3]). By this theorem, the sequence

$$(2\log\log n)^{-1/2}S_{i,n} \quad (n=1,2,\ \dots\ )$$

is relatively compact in the space $C[0,1]$ and the set of its limit points coincides with the set $\mathcal{S}$ of absolutely continuous $s\in C[0,1]$ such that $s(0)=0$ and $\int_0^1 (s'(t))^2\,dt \leq 1$.

Consequently

$$\limsup_{n\to\infty} \Big/ \liminf_{n\to\infty} \Big/ n^{1/2}(2\sigma^2 \log\log n)^{-1/2}\left(n^{-1}T_{kn}-\mu\right) =$$

$$= \limsup_{n\to\infty} \Big/ \liminf_{n\to\infty} \Big/ \tau_k\left((2\log\log n)^{-1/2}S_{1,n}, \dots ,(2\log\log n)^{-1/2}S_{k,n}\right) =$$

$$= \sup \Big/ \inf \Big/ \{\tau_k(s_1, \dots ,s_k): s_1, \dots ,s_k \in \mathcal{S}\}$$

If $s\in\mathcal{S}$ and $0\leq t_1\leq t_2\leq 1$ then by Hölder's inequality $s(t_2)-s(t_1)\leq(t_2-t_1)^{1/2}$. Thus

$$\sup \{\tau_k(s_1, \dots ,s_k): s_1, \dots ,s_k\in\mathcal{S}\} \leq$$

$$\leq \sup \Big\{ \sum_{i=1}^{k} (t_i-t_{i-1})^{1/2}: 0=t_0\leq t_1\leq \dots \leq t_{k-1}\leq t_k=1\Big\} = \sqrt{k}$$

and this value is attained when $t_i=i/k$ and

$$s_i(t)= \begin{cases} 0 & \text{if} \quad 0\leq t\leq t_{i-1} \\ \sqrt{k}(t-t_{i-1}) & \text{if} \quad t_{i-1}<t<t_i \\ 1 & \text{if} \quad t_i\leq t\leq 1 \end{cases} .$$

On the other hand

$$\inf \{\tau_k(s_1, \dots ,s_k): s_1, \dots ,s_k\in\mathcal{S}\} \geq \inf \{s(1): s\in\mathcal{S}\} = -1 ,$$

which is attained when $s_1(t)= \dots =s_k(t)=-t$. This completes the proof. □

5. The opposite question

In this section we shall deal with the following problem of opposite type: Where do the packets of the message stay in the communication system at a given moment?

Since the waiting time of the ith packet at the source node is not less than $i-1$, therefore we shall denote by U_{ij} the number of steps it makes during the first $j+i-1$ periods.

Let Y_{ij} $i=1,2, \dots ,k;\ j=1,2, \dots$ be independent identically distributed indicator variables with $P(Y_{ij}=1)=p$. The following relations are similar to (1):

$$U_{1,j}=Y_{1,1}+Y_{1,2}+ \dots +Y_{1,j}$$

$$U_{i,j}=(U_{i,j-1}+Y_{ij})\wedge U_{i-1,j} \qquad U_{i,0}=0 \qquad i\geq 2$$

By induction it can be shown that

$$U_{kn} = \min \{ \sum_{i=1}^{k} \sum_{j=j(i-1)+1}^{j(i)} Y_{ij} : 0=j(0)\leq j(1)\leq \dots \leq j(k-1)\leq j(k)=n\} =$$

$$= - \max \{ \sum_{i=1}^{k} \sum_{j=j(i-1)+1}^{j(i)} (-Y_{ij}): 0=j(0)\leq j(1)\leq \dots \leq j(k-1)\leq j(k)=n\}$$

Consequently

$$n^{-1/2} | U_{kn} + T_{kn} - Y_{ij} | \to 0 \qquad \text{a.s.},$$

in other words, the opposite problem leads to the same mathematical situation as the initial one. Thus from the preceding Propositions limit results can be obtained for the asymptotic location of the packets in the network, keeping in mind that it is $U_{k,n-k+1}$ which gives the position of the ith packet after n periods.

PROPOSITION 5

a) $\lim_{n\to\infty} n^{-1} U_{k,n-k+1} = p$ *with probability 1,*

b) $(\hat{U}_{1,n}, \hat{U}_{2,n}, \dots, \hat{U}_{k,n}) \to_d -(W(1), \tau_2(W_1, W_2), \dots, \tau_k(W_1, \dots, W_k))$

where $\hat{U}_{i,n} = n^{1/2}(p(1-p))^{-1/2}(n^{-1} U_{i,n-i+1} - p)$. □

We remark that the results and proofs of this paper can easily be carried over the case when the probabilities of "free" states differ for different channels. Naturally this probabilities p_i must fulfil some conditions of Lindeberg type, e.g. for the extension of Proposition 3 the condition

$$(\min_{1\leq i\leq n} p_i^2) \sum_{i=1}^{n} (p_i^{-2} - p_i^{-1}) \to \infty \qquad \text{as} \quad n\to\infty$$

will do. Details are left to the reader.

6. Open problems

Finally we mention some open problems which are worth dealing with.

It would be of independent interest to investigate the asymptotic distribution or at least the expectation of $\tau_k(W_1, \dots, W_k)$, when k tends to infinity. The best estimate I know at this moment is the following:

$$C_1 k^{1/2} \leq E\tau_k(W_1, \dots, W_k) \leq C_2 (k \log k)^{1/2}$$

The lower estimate is obtained by imposing further restrictions on the partition of $[0,1]$ in the definition of τ_k:

$$\tau_k(W_1, \dots, W_k) \geq \sup \{ \sum_{i=1}^{k} (W_i(t_i)-W_i(t_{i-1})) : t_0=0,\ t_k=1,\ \frac{i-1}{k-1} \leq t_i \leq \frac{i}{k-1} \} =$$

$$= (k-1)^{-1/2} \sum_{i=1}^{k-1} \tau_2(\hat{W}_{2i-1}, \hat{W}_{2i})$$

where $$\hat{W}_{2i-1}=(k-1)^{1/2}(W_i(\frac{i-1+t}{k-1})-W_i(\frac{i-1}{k-1}))$$

$$\hat{W}_{2i}=(k-1)^{1/2}(W_{i+1}(\frac{i-1+t}{k-1})-W_{i+1}(\frac{i-1}{k-1})) \quad , \qquad i=1,2, \dots, k-1$$

that is $\hat{W}_1, \hat{W}_2, \dots, \hat{W}_{2k-2}$ are independent standard Wiener processes.

For the upper estimate notice that if the intervals $[t_{i-1}, t_i]$ are arranged in decreasing order, then the length of the ith one does not exceed $1/i$. Therefore

$$\tau_k(W_1, \dots, W_k) \leq \sum_{i=1}^{k} \bigvee_{j=1}^{k} \sup \{W_j(t)-W_j(s): 0 \leq s \leq t \leq 1,\ t-s \leq 1/i \}$$

On the right-hand side for fixed i

$$\sup \{W_j(t)-W_j(s): 0 \leq s \leq t \leq 1,\ t-s \leq 1/i\} \leq$$

$$\leq 2 \bigvee_{r=1}^{i} \sup \{|W_j(t)-W_j(s)|: \ t,s \in [\frac{r-1}{i}, \frac{r}{i}] \} =$$

$$= 2\, i^{-1/2} \bigvee_{r=1}^{i} (\sup_{0 \leq t \leq 1} \hat{W}_{jr}(t) - \inf_{0 \leq t \leq 1} \hat{W}_{jr}(t))$$

where

$$\hat{W}_{jr}(t)=\sqrt{i}(W_j(\frac{r-1+t}{i})-W_j(\frac{m-1}{i})) \quad j=1,2, \dots, k;\ r=1,2, \dots, i$$

are independent standard Wiener processes.

Taking expectation it follows that

$$E\tau_k(W_1, \ldots, W_k) \leq 4 \sum_{i=1}^{k} i^{-1/2} \bigvee_{j=1}^{k} \bigvee_{r=1}^{i} \sup_{0 \leq t \leq 1} \hat{W}_{jr}(t) =$$

$$= O\left(\sum_{i=1}^{k} i^{-1/2} (\log ik)^{1/2}\right) = O((k \log k)^{1/2}).$$

It is another interesting problem to characterize the asymptotic network delay in the case when k is not fixed but it tends to infinity with n.

References

[1] KLEINROCK, L.(1976): *Queueing Systems. Vol.II. Computer Applications.* Wiley, New York

[2] BILLINGSLEY, P. (1968): *Convergence of Probability Measures.* Wiley, New York

[3] STOUT, W. F. (1974): *Almost Sure Convergence.* Academic Press, New York

REDUCTION OF SPACE REQUIRED BY A LARGE CIVIC REGISTRATION SYSTEM

V. OLÁH - ZS. PINTÉR - J. P. SÜTŐ

Research Institute for Applied
Computer Sciences
H-1536 Budapest,
P.O.B. 227

ABSTRACT

The design of a file and record system generally involves several major problem areas. One of the main objectives is the reduction of storage space.

In this paper, a record partitioning technique is presented related to and used in a special type of application.

The file system considered stores millions of records and is maintained on secondary storage devices. Savings in storage space could be realized by partitioning the fields of a record into a basic and an optional record segment.

Using an optimal record segmentation in the redesigned file system, the achieved savings in space amounted to more than 40 percent.

INTRODUCTION

In Hungary, a population registration system has been under development and partly in operation for some years.

The data base on a HwB 66/60 computer stores information about 10 million individuals. The file system is organized into indexed sequential files processed by the Indexed Sequential Processor software /ISP in the sequel/ and is maintained on more than forty disk-volumes, each of capacity of one hundred Mbytes. Each record containes a predetermined set of 60 data items of fixed length.

Because of configuration limitations we cannot have online access to the whole system. In addition, the large number of disk-volumes causes several maintenance and operation problems. In order to overcome these drawbacks we were seeking possible forms of reducing the storage space and thus the number of disk-volumes holding the data base.

1. CLASSIFICATION OF DATA ITEMS

To achive our main objective, we classified the data items of a record as falling into one of the following categories:

A/ items with encoded contents,
B/ items not applicable for encoding,
C/ items, which are rarely used.

Some data items, such as the Christian names or home addresses have been recorded in coded form since the early days of the system. This needed setting up code-tables and required coding and decoding functions to perform, but it also gave the opportunity to check and validate or even correct input data.

The correspondance between these data and their encoded equivalents has been established in so called directories.

The idea of directories meant the first attempt towards space reduction. Even with the home address directory alone savings were significant, since a home address could be compressed into 6 characters, while an average home address could have taken up 26 characters otherwise. More details about this directory were presented by L. Hartwig at the 5th Conference on the Theory of Operating Systems, Visegrád [1]

To build up directories for other fields of a record, such as surnames or places of birth, wouldn't have been economic or even possible because of the practically infinity number of chances. Therefore, a maximal number of characters had to be reserved for them. For example, a thorough examination of Hungarian surnames showed that they do not exceed 23 characters in length. Since married names are postfixed by an additional two characters, a total of 25 characters were needed for this field. For similar reasons, the birth-place field length was set to 22 characters which is long enough to hold any town or village name.

As to the items in category C, we note the following. After the system is in operation for some time, the system designer can turn to such problems as real space utilization and frequency of updates of a data item. If several data items make up a record, it may happen that certain fields in a particular record have no contents at all, or their contents occupy less space than is allowed. In a system with a large number of records the space such wasted may become significant. Our experimental results showed that some fields remain empty or unaffected during updates in at least 90 percent of the records. An example for such a field is the temporary home address.

Having these information on the working system, we are left with the problem of reducing the unused space. This can be done by redesigning the file and record system. Care must be taken, however, not to have a higher processing time at the cost of the new structure.

2. NEW RECORD STRUCTURE

The existing data base was organized as an indexed sequential file system with fixed-length records. The new system will contain variable-size records to provide greater flexibility.

Three segments will constitute a record.
Those fields which will always be present in every record are contained in the first segment and are called basic fields. The rest will make up the third segment and are called optional fields. They may or may not be present, as indicated by the set of flags in the second segment. If the ith flag is set, the ith optional field is part of the record and is assigned space in the third segment, otherwise the ith optional field has no contents and occupies no space in the specific record realization.
A basic field, if present, is not assigned the total space it would require, but it is stored according to the following rule: A space of length k / k will be determined in the next section / is always reserved for it. If the actual contents fit into this fixed-length space, a space of length k is allocated to it and there is no need for additional space for its extention in the third segment. If the predetermined length does not suffice, the excess over k is stored in

the third segment as an optional field.
Examples for basic fields are the surname and birth-place fields.
Our objective now is to determine the basic field length k so as to minimize the total storage space occupied by these fields.

3. DETERMINATION OF BASIC FIELD LENGTH

We introduce a discrete random variable ξ, whose definition will depend on the type of field in consideration. ξ takes up the value x, if the actual field contents is x characters long. A distinction will be made as to the range of values of ξ: it will be taken as $1,\ldots,25$ for surnames and $1,\ldots,22$ for birth-places. Accordingly, the maximal field length, denoted by K, will either be $K=25$ or $K=22$.
We define the theoretical distribution function of ξ as

$$F(x) = P(\xi < x), \quad x \in (-\infty, \infty)$$

and the empirical distribution function by

$$F_n(x) = \begin{cases} 0 & x \leq \xi_1^* \\ \frac{i}{n} & \xi_i^* < x < \xi_{i+1}^* \\ 1 & \xi_n^* < x \end{cases}$$

where n denotes the sample size and $\xi_1^*, \ldots, \xi_n^*$ is the ordered sequence of the observed values of ξ / thus, $F_n(x)$ is the frequency of the first n observations less than or equal to x /.

We now face the following optimization problem.
We have to minimize the function

$$(1) \qquad H(k) = k + (1 - F(k))(K - k)$$

where k varies from 1 to K.
Clearly, $H(k)$ is the expectation of the field length.
Minimization of $H(k)$ is equivalent to minimization of

$$h(k) = F(k)(K - k).$$

The minima will give the optimal length k^* of a basic field.
In practice, one can, of course, only use $F_n(k)$ instead of $F(k)$ and compute the value $F_n(k)(K-k)$ instead of $F(k)(K-k)$.

The GLIVENKO-CANTELLI theorem, however, asserts that

$$D_n = \sup_x |F_n(x) - F(x)| \xrightarrow{n} 0 \qquad \text{with probability one.}$$

While this reassures the statistician that, for large n, F_n should be a good estimate of F in the sense of making D_n small, he cannot infer, say, how large n should be in order to insure that

$$P(D_n \geq \varepsilon) \leq \delta$$

where ε, δ are two pre-assigned arbitrary positive numbers.

A well known result of SMIRNOV, restated below, helps us with determining the sample size n. / $F(x)$ and $F_n(x)$ denote the distribution functions as above /.

THEOREM / SMIRNOV [2] /:

$$(2) \qquad \lim_{n\to\infty} P\left(\sup_{-\infty<x<\infty} (F_n(x) - F(x)) < \frac{y}{\sqrt{n}}\right) = \lim_{n\to\infty} P\left(\sup_{-\infty<x<\infty} (F(x) - F_n(x)) < \frac{y}{\sqrt{n}}\right) = 1 - e^{-2y^2}, \quad y \geq 0.$$

It should be noted that in our case, n is to be chosen so as to have

$$K\left[F_n(k) - F(k)\right]$$

sufficiently small. Since a field contains an integer number of characters, it is enough to hold the following relation:

$$(3) \qquad K\left|F_n(k) - F(k)\right| < 0,1 .$$

If we set $1 - e^{-2\gamma^2} = 0,95$ and carry out the necessary calculations using (2) and (3), we have for n, $n \approx 150 \cdot K^2$.

Machine calculations, based on a sample of size of 75 thousand records, showed that a choice $k^* = 8$ for surnames and $k^* = 11$ for birth places must be made.

4. FURTHER DISCUSSIONS AND REMARKS

The optimal values for the basic field lengths, obtained in the preceding section, have no relation to the operating environment. In a real-life system, however, this must also be taken into consideration.

Our application runs on a HwB 66/60 computer and makes use of its indexed sequential file processing technique, ISP, which handles variable length records.
A feature of ISP is that if a longer record is to replace the existing one during update, the new record is placed on overflow area and the space into which the updated record did not fit, is marked inactive but fisically deleted only after rebuilding the ISP-file.
In light of this, the optimal basic field lengths had to be modified in order to avoid frequent overflow writes. The future

system will use 12 characters long surname fields and 16 characters long birth-place fields.

Statistical investigations were carried out also for the space required by each optional field within a record. It was discovered that more than the half of the records did not utilize any optional field. The expected optional field length proved to be 26 percent of the total length of the optional fields. Having computed the average space needed for a redesigned record as

total basic field length +
6 characters for the flags +
expectation of the length of the optional fields

we concluded that using such a record segmentation, about 40 percent storage space reduction could be incurred.

REFERENCES

[1] L. Hartwig: "Experiences with Honeywell's IDS/I" lecture, given at the 5th Conference on the Theory of Operating Systems, Visegrád, 1980. febr. 28-30

[2] A. Rényi: Probability Theory /in Hungarian / Budapest, 1968.

EFFICIENCY PROBLEMS FROM THE EDP DESIGNER POINT OF VIEW

Adam Peterseil
Marek Parlewicz
Włodzimierz Ptak

ZETO - Wrocław, Poland

1. INTRODUCTION

The paper presents an application of a simple method of programs run time and resource loadings prediction. The method allows for analysing both mono and multiprogrammed processing and is implemented as an easy - to - use software packet. It aims to be a helpfull tool in EDP systems desing process.

A program run time and corresponding resources loadings are of the most importance among other parameters describing an EDP system. There parameters are often taken as decision criteria and at least aid the decision by taking other quality factors, e.g. portability, flexibility easiness to handle etc. in a choice of the possibly best EDP system solution.

The objects of the evaluation performed by the packet are DP application systems and such the evaluation is assumed to be on of the design stages mode by the designer after fulfilling higher order perequisites and constraints. The aim of such the evaluation is to achieve possibly short run or transaction turnaround times by configuration adjustment, proper data files organisation and accessing avoiding bottle-nacks.

General steps of the packet use are as follows:
- define inputs
- calculate measures
- evaluate results, repeat if unsatisfactory with the changed input definition.

2. PACKET CONSTRUCTION

There are some languages of a packet allowing designer to define his problem.

The languages are as follows:
- program description language /PDL/
- configuration description language /CDL/
- operating system description language /OSDL/
- multiprogramming mix description language /MDL/.

On the basis of problem description the packet calculates program run time and corresponding resources loadings on a given machine with a given operating system, both in mono and multiprogrammed environment.

Monoprogram evaluation is made by simple calculations, anyhow gives enough accuracy. Multiprogrammed evaluation is performed by the hybrid simulation - analytical method and uses as an input the results of one - program analysis. Simulation is used to map time passage and to indicate programs actually being multiprogrammed. The analytis part calculates the ratio of the run time increment due to multiprogramming. Functional structure of the packet is shown on the fig.1.

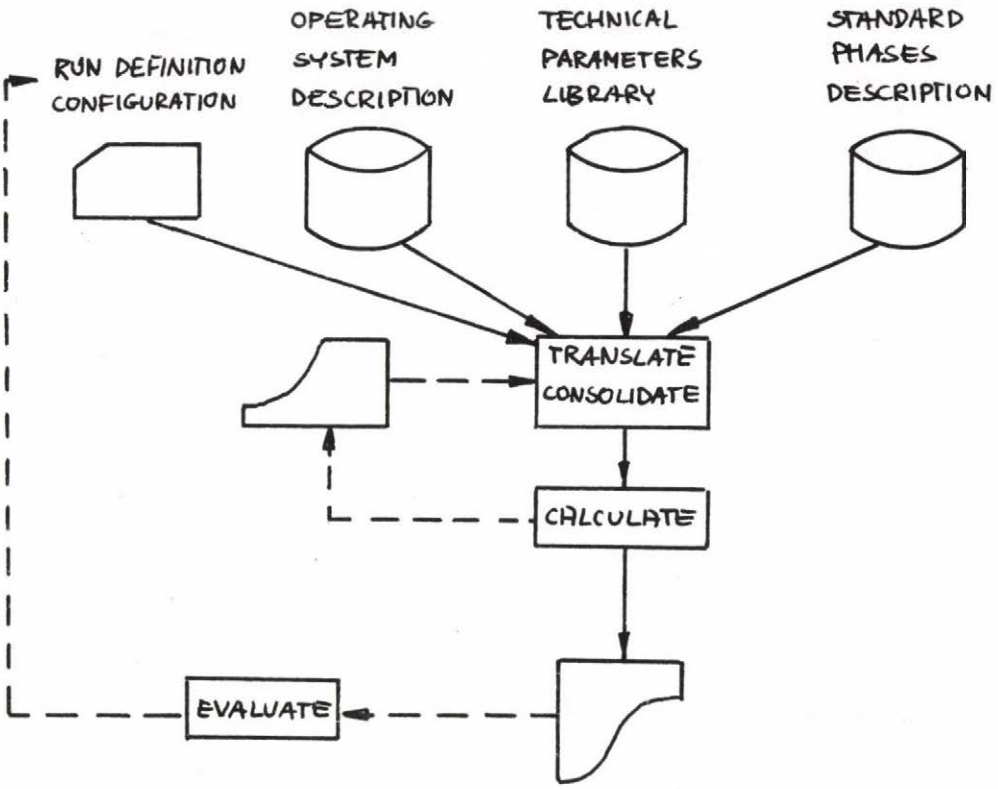


Fig. 1. FUNCTIONAL STRUCTURE OF THE PACKET.

Program description language allows for defining any DP run /key word RUN/ in the terms of programs /key word PROGRAM/. Each of the programs consist of so called phases /key word PHASE/ general feature of wchich is that devices are assigned to the phase at the beginning and released at the end. Finally, each of the phase consists of the chains /key word CHAIN/, each describing the unique algorithm performed by the program on a record /by GET record, PUT record and EXEC elements/. GET and PUT elements describe the file from /to which the record is read/ written by file allocation and its name, EXEC denotes the record processing time in ms.

One can also use the description of standard phasses or programs which are in the library. At the current stage the SORT programs are included. TO illustrate above - see figures 2 and 3.

```
RUN  name
   PROGRAM  name
     PHASE  iteration counter
       CHAIN  percentage
         elements          — { GET  device, file
                               PUT   "     "
                               EXEC  processor time
   PROGRAM  name
     PHASE  STANDARD
     name
   FILES
     name & parameters
     :
CONFIGURATION
   PROCESSOR  name or parameters
   CORE  parameters
   devices                 — names or technical parameters
```

Fig. 2. STRUCTURE OF PROGRAM DESCRIPTION LANGUAGE (PDL) AND CONFIGURATION DESCRIPTION LANGUAGE (CDL).

```
MULTIRUN  MIX
   RUN  name
     PROGRAM  name
       PHASE  number
       devices utilization   — { device, percentage
       START TIME  time         :
       PHASE EXECUTION TIME     :
       PHASE
       :
     PROGRAM
     :
OPERATING SYSTEM
name or parameters
```

Fig. 3. STRUCTURE OF MULTIPROGRAMMING MIX DESCRIPTION (MDL)

The separate part is the files description /key word FILES/ describing files allocation, structure, accessing and blocing.
The files allocation is connected with the configuration description /key word CONFIGURATION/ i.e. structure and devices used within /key word Unn in the case of unit-record devices and Unn; Umm in the case of stores - Umm elenotes the controller/ via devices numbers /nn/.
One can either describe devices technical parameters in the above description or merge them from the library using only the names.

Operating systems description uses only the names, the parameters are taken from the library /the current version of the packet includes only houskeeping activity/.
Multirun mix description allows for defining any tree-structured schedule of runs.
The run definition is taken from one - program analysis and contains run times and resources loading of each phase.
It allows also for any background load description /in the terms of the devices loadings/.

3. THE EXAMPLE

The section describes the simple example of the packet application.

Two runs are considered:

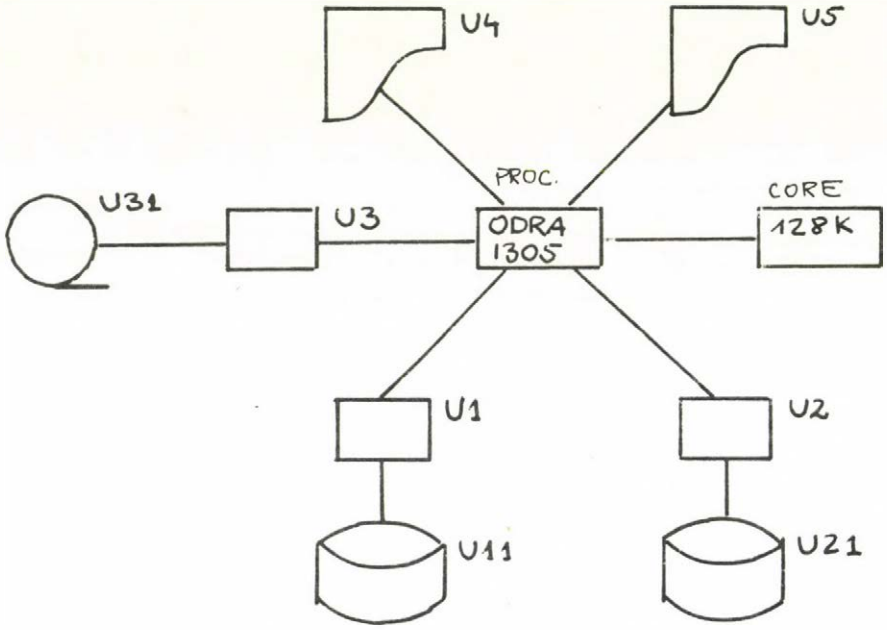
- A1 - with program UPDATE
- A2 - with program SPOOLER

in mono-program analysis.

The configuration considered is shown on the fig. 4, A1 and A2 runs on the fig. 5 and 6, respectively.

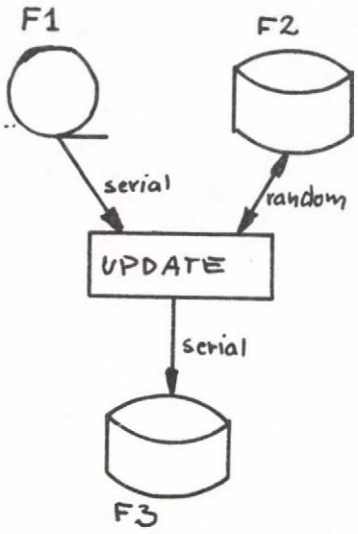
Multi-run analysis consists of A1 and A2, A3 multirum mix. /see fig. 7/. A1 - update, A2, A3 - two spoolers/.
Each of the runs begins at "0" instant.

Multirun definition includes the results of one - program analysis, so one can find that A1 run lasts 66,25 s and A2 - 108 secs /see also resources loadings /.
On the fig. 8 one can find the multi-run analysis results. The total processing time is 122.04, with the increment of A1 run time of ca 8s and A2 - of 14 s.
Corresponding resources loadings one can obviously achieve.



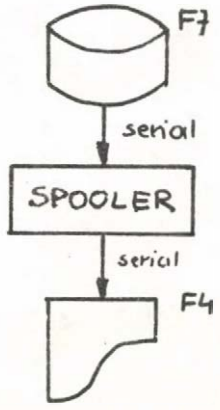
CONFIGURATION
PROCESSOR ODRA 1305
CORE 128K
U1: EDS325
U2: EDS325
U3: MTS304
U4: DW325
U5: DW325
U11: U1: EC5052
U21: U2: EC5052
U31: U3: PT3M

Fig. 4. CONFIGURATION. DIAGRAM AND DESCRIPTION IN **CDL**.




```
RUN A1
PROGRAM UPDATE
PHASE 500
CHAIN 100
GET * MT31, F1
EX 2.5
GET * DA11, F2
EX 7.5
PUT * DA21, F3
PUT * DA11, F2
FILES
F1:= BL:512, BLF:5, DVOL:500
F2:= BUCK:512, BLF:4, DVOL:500, RANDOM
F3:= BUCK:1024, BLF:5.12, SERIAL
```

Fig. 5. PROGRAM UPDATE.
DIAGRAM AND DESCRIPTION IN **PDL**.




```
RUN A2
PROGRAM SPOOLER
PHASE 500
CHAIN 100
GET * DA21, F7
PUT * <P4, F4
FILES
F4:= BLF:0.33
F7:= BUCK:1024, BLF:5.12, SERIAL
```

Fig. 6. PROGRAM SPOOLER.
DIAGRAM AND DESCRIPTION IN **PDL**.

```
MULTIRUN MIX
RUN A1
PROGRAM UPDATE
PHASE 1
PROCESSOR: 12.63
U3: 4.23
U31: 4.23
U1: 47.17
U2: 7.40
U11: 100
U21: 7.84
PRT: 66.25 SECS
RUN A2
PROGRAM SPOOLER1
PHASE 1
PROCESSOR: 4.10
U21: 9.58
U2: 9.04
U4: 100
PRT: 108 SECS
RUN A3
PROGRAM SPOOLER2
PHASE 1
PROCESSOR: 4.10
U21: 9.58
U2: 9.04
U5: 100
PRT: 108 SECS
OPERATING SYSTEM: E6RM
END
```

Fig. 7. THREE-RUN MIX: A1, A2, A3. DESCRIPTION IN **MDL**.

```
TOTAL PROCESSING TIME: 122.64 SECS
RUN A1 : 74.86
RUN A2 : 122.04
RUN A3 : 122.04
DEVICES LOADINGS:
:
:
:
```

Fig. 8. EXAMPLE OF RESULTS. (corr. to fig. 7.)

4. CONCLUDING REMARKS

The described packet is opened in the sense of the possibility of including new devices, operating systems and standard programs into the library.
The future development is assumed.

Finally it is worth nothing that the packet is used on a trial - and - error basis and efficiency of its use depends primarily on the creative activity of the designer.

5. REFERENCES

1. Practical sizing, ICL Training and Education Publication.
2. Sizing for Systems Designers, -"- -"-
3. Kerner H., Beyerle W., A. Simulator Generator Based on Formal Description of Architectural, Load and Operating System Model,TU Vienna, 1 78.
4. Błaszczyk Z., Peterseil A., An Approach to Performance Evaluation and Management of a Time - Sharing Computer Installation by Analytical Methods, in: Proceedings of the 5-th Conference on Operating Systems, Visegrad, 1979.

High Level Computer /Network/ Design Proposal based on Open Systems Interconnection

PETER POKA
Computing and Automation Research Institute
of the Hungarian Academy of Sciences
Budapest, Postafiok 63; 1052; Hungary

The High Level Computer /Network/

In the sixties, when Data Communication Networks were already commonplace, a new kind of communication method was proposed /1/ to meet the extreme reliability and resiliency requirements of military voice communication. The implementation of this method - called packet switching - has been necessitated the use of small computers as building blocks and hence the new networks have been called Computer Assisted Data Communication Networks. Further investigation showed these communication networks particularly suitable of handling the bursty pattern of traffic between cooperating data processing system. And such the term "Computer Network" started to mean an ensemble of cooperating DP systems /host COMPUTERS/ and a well detached data communication subNETWORK /consisting of communication channels, lines and a variety of computing instruments e.g. mini computers/. These networks can be classified by the level of their interactions. The first class consists of systems providing service for the end users /terminal networks/ and the second group corresponds to the Value Added Networks in which the cooperating host system have higher grade /not only terminal fitting/ intersite data traffic among each other. When the first function is supported inside a second type net we call the network Generalized Value Added Network /GVAN/. One should notice that the programmers are always working on the level and with the terms of the abstracted object shown by the architectural description /2/ of the computational device. If this picture shows a minutely resemblance to the real instrument, we have the case of assembly -like device and programming. If the projected tool doesn't show this close resemblance to the real

life device, but rather uses terms known from non - computational human activities, in connection with the problem we call the total system, the description method and the applied tools high level.

The introduction of a device, programmable in HIGH LEVEL languages architecturally dressed up like a COMPUTER and implemented with the assistance of a data communication sub-NETWORK hides the structure and covers the very existance of the networks from the users. This logical device /called High Level Computer/Network/ -HLC/N/ / can show the user defined interface over a very broad, and constantly changeable domain of the components /heterogeneous host machines, data communication facilities etc./ and constituted by the coordinated interacting activities of subsystems selected time dependently from the currently available servicepoint set. Here the network isn't a final product but rather an implementation tool as it's indicated by the bracketing of the term "network".

Open Systems Interconnection

One can easily enumerate many problems which can be more economically computed if they were partitioned and physically distributed. Because the run-time distribution has distinct superiority over the compile-time one in respect of reliability and availability, the optimal solution of the distribution problem is certainly a run-time phenomenen.

The full technical burden of the creation of geographically distributed run-time systems should and can not be put onto the users, hence the HLC/N/ itself is proposed to be implemented as a distributed data processing System /7/. By this concept the only distributed /application/ problem is the driving of the HLC/N/ which in turn - owing to its internal attributes - will serve the geographically dispersed computational needs.

A new "inherently communicative" bred of supervisors, running in HLC/N/'s constituent parts will obvite the former necessity of writing distinct communication user programs with the appropriate rum time support packages. These supervisors should support an

HLC/N/ - wide message based inperprocess communication /4/, hiding the run-time attributes /e.g. distance, locality/ of processes; and a dynamic process creation capability, invokable by all sorts of external sognals /unexpected terminal interrupt, remote supervisor message etc,/.

The currently emerging ISO work /3/, called Open Systems Interconnection /OSI/ addresses the problems of Systems Consisting of Computers, peripherals, telecommunication devices etc. which are interrelated by information exchange. The document formalizes the distinction between architectural and structural aspects - being the first of the vierwer's or the user's and the second of the builder's sight - puts partial problems into prespective and proposes a common reference language. Above these it attempts to create a frame which can support the user's efforts for the symbolic definition and layered implementation of distributed applications even with the present hostile heterogeneity of operating system environments. OSI with its coherent view gives a guildeline to every user to build partitioned /by geographical locality/ systems in a way which gives the cost optimal resiliency against the historical or relocation triggered abrupt modification of the borderline between Application Objects and Supporting Environment. The distributedness and communicative cooperativity of HLC/N/ shift the emphasis back to COMMUNICATION from computing and turn the system programmer's attention to OSI, a possible structural design and reference method of the constituent system parts.

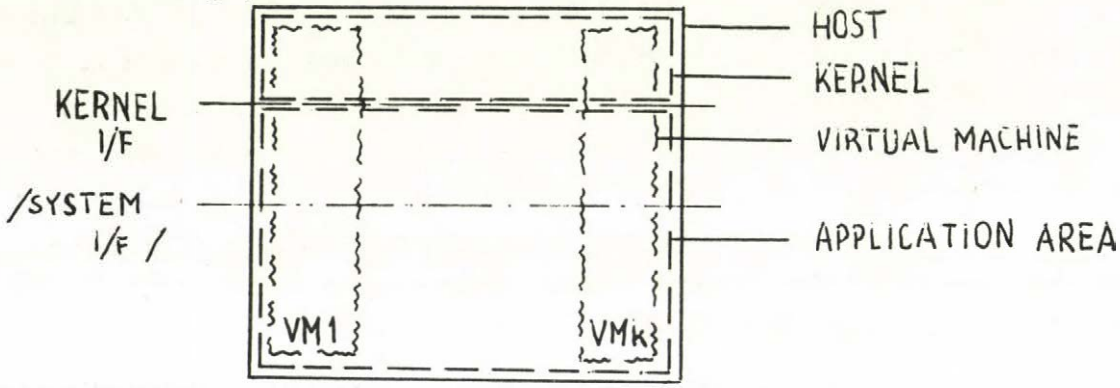
Draft System Proposal

The System - as it was first proposed in /2/ - shows the architecture of a high level language machine and serves the ad hoc connections and activities of passive and active informational entities /users, files, hardware, devices etc./. These entities are represented by processes and the system can be viewed as a cooperating society of the sequential processes which progress in parallel, with undefined speed ratios.

The processes use a message based interprocess communication mechanism /4/, /5/, which is

- universal: used by all kind of processes
- transparent: hides the actual run time conditions /e.g. physical distance/ from the cooperating processes
- distributed: works according to such an algoritm

The internal structure of the system at each host location showed on the following picture



Here the KERNEL comprises the time critical functions of the local system /e.g. interrupt analysis/ the necessary support and administration services for driving the hardware resources /CPU, memory, 'other' perpheral devices etc./ and in cooperation with other KERNEL-s provides the message service for the Application Area processes /Application Entities in OSI term/.

The KERNEL contains the five lowest layers of OSI Reference Modell and controls the message passing in the Session Layer, whose purpose is the support of interactions between cooperating application Entities /AE/. A SESSION will be a communicative cooperative relationship /CR/ between two AE-s characterising their communications. If one considers the PORTs of processes /4/ utility communication devices /symbolicly identified by logical I/O channels the session can be regarded as a pairing of two ports or two SESSION ENTITIES /SE/. The Session Layer is composed of a pool of unique ports and a housekeeper routine /HKR/, which actually builds the CR-s cooperating with other HKRs via permanently open and dynamically assigned CR-s /called command lines/. The HKR receives calls from

AE-s, allocate a communication device /port or SE/, keeps account of the AE/SE pairs and as a TRANSPORT USER initiates appropriate actions via the TRANSPORT SERVICE. At the other and a SE will be selected and made responsible for activating or priming the AE identified by the AE address in the original call.

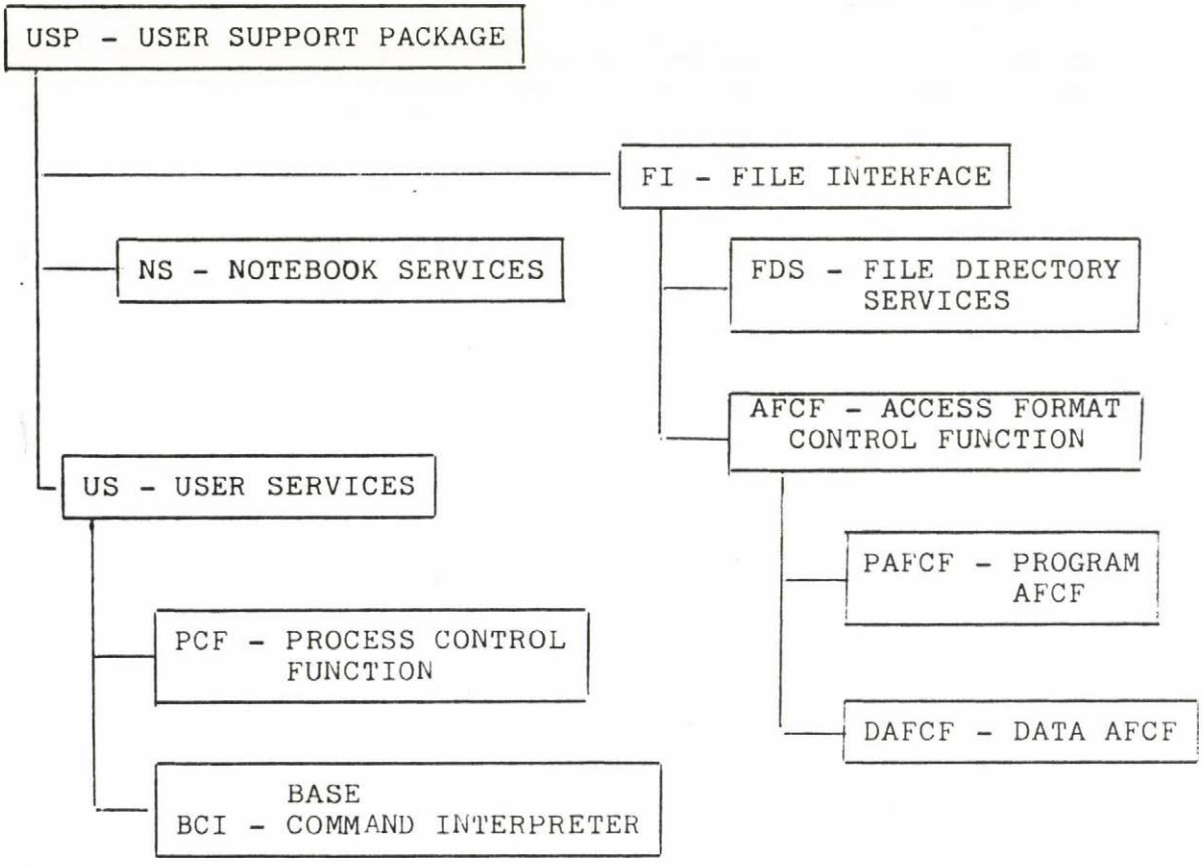
At present the NETWORK LAYER /NEL/ is based upon the use of telecommunication facilities. One can generalise this to other transportation means and special hardwired information sources and sinks. The data communication subnetwork can be treated as a special distributed firmware peripheral device of the system, dedicated to the support of message switching and NEL as a special part of a general DEVICE LAYER /DEL/ which drives virtual devices via abstract links.

Even the EVENT Control functions of the KERNEL which handle the interrupt system and the privileged operations' section - admittedly the most closely driven virtual peripheral device - can be conceived within this frame /6/. The Event Recogniser /ER/ part, a coordination message receiver has a Physical Layer section consisting of hardwired connections and control , a set of dedicated buffers /registers/ and semaphores /Interrupt Flag Register - IFR/ which are dinamically assigned to higher level entities by the hardware event initializers. In systems where the ER doesn't produce the messages in compatible format with other system messages, but leaves them in registers and performs a V-operation on IFR it's necessary to run a message converter software routine /P-operation on IFR, convert and queue up message, V-operation on message semaphore/ to eliminate the differences. The rest of the Physical Layer is made up of the Even Initializer sections / privileged I/Θ, program control, set mask instructions/ of the generalized device drivers. In present day systems the implementation of the message service is built upon a multilevel queueing system where the IFR /a collection of binary hardware semaphores/ is supported by a system of software semaphores /SVC calls/ and a certain portion of them is dedicated to "well known service point"-s. This method which requires only the caller's full identification, slightly deviates from the general solution but significantly speeds up the overall service of the requests.

Between the KERNEL and the Application Area can be found a rigidly enforced boundary /called KERNEL Interface/ which can only be crossed via interrupts. The Application Area is occupied by processes - Application and Presentation Entities in OSI terms- and can be divided into three conceptual subarea

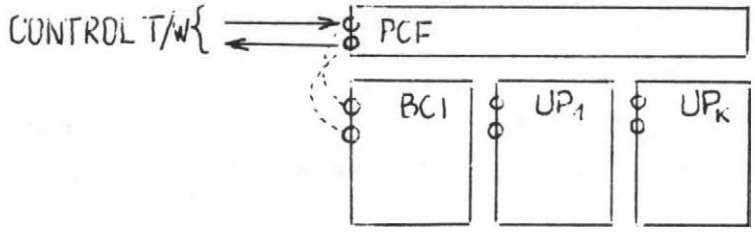
System Support Are /SSA/
User Support Area /USA/
User Work Area /UWA/

The System Support Package - run in SSA - comprises the non time critical functions of the system such the file system and the consol communication service etc. The User Support Package - run in USA - has the following internal structure



Here the User Services are primarily concerned with the administration of the command sources and the execution of user Processes.

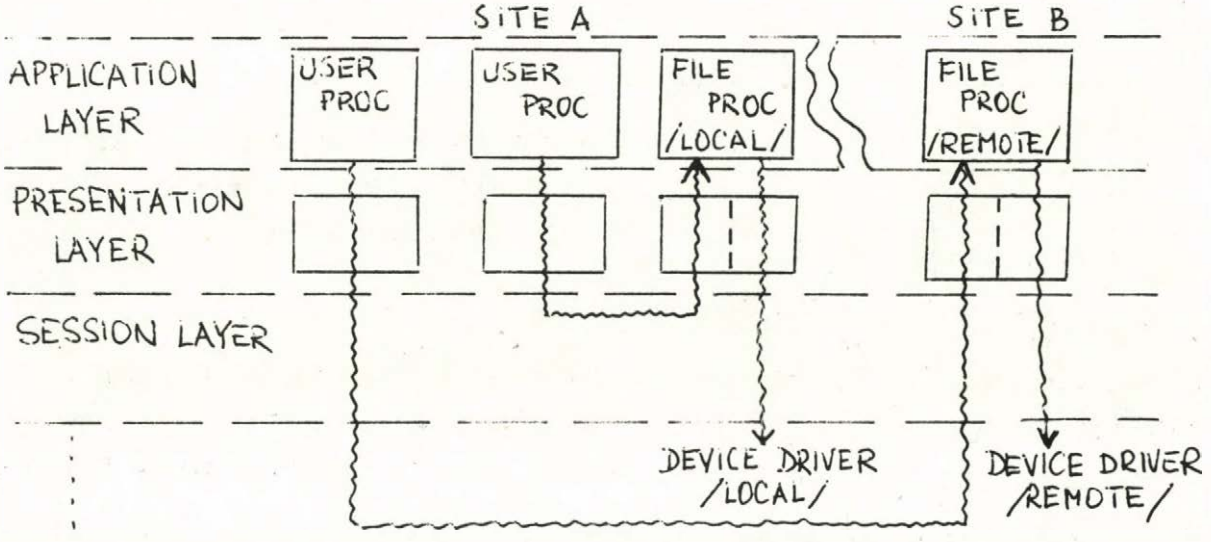
PCF is basically a control envelope of the User Processes - run in UWA - and of the BCI which handles the user's base command language.



Conceptually PCF owns the connection to the control typewriter and passes it to the activated BCI - the system is in COMMAND state - or to the appropriate user process. It deals with contingencies /program interrupts and signals from the system/ and is responsible for the initation, management and termination of processes within the virtual machine. It is callable from processes via routine calls or from the consol typewriter via the interrupt sequence. The BCI enables the user to submit commands on his console in a form of standard procedure call in BPL. This consists of the name of an external routine followed by the list of arguments. The insertion of new commands is fairly easy, because it needs only the translation of the proper routine. In fact external routines with string parameters can be used as foreground commands.

The NS is primarily concerned with recording of the correspondence between the formal and actual file parameters and providing facilities for communication between USP components.

The FI provides the appropriate connection to the File System which accomodates arbitrary memory images in a standard file format, maintains a library structure for each user and serves UWA members through file processes as the following picture shows.



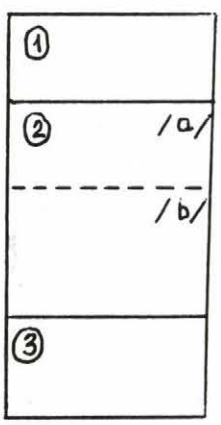
These processes completely hide the internal structure of the files and give or receive information via ordinary interprocess message communication. Opening a file corresponds to the creation of a file process which will be killed at closing time.

FDS consists of the routines which activate the File Directory manipulation services. A partial list of these might be:

CONNECT FILE
DISCONNECT FILE
CREATE FILE
DESTROY FILE
SET/GET FILE INFORMATION /which can be fanned out for
size, archiving status, access permission,
access mode, name etc./

AFCF is composed of Presentation Entities and provides the access or communication mechanism to the File System. In PAFCF for instance the compilers use a standard code generation routine, pass the information to it in unordered sequences and the file in fact is reconstructed by the generation routine. In the opposite direction the LOADER can be regarded a special presentation agent of the call session.

The UWA is the designated place to run the problem oriented programs - all are application Entities - like compilers, editors, utilities or the user provided routines. UWA like the rest of the system has a pure procedure environment made up of recursively callable routines. The core format of the processes shows the following layout.



1./ CODE SECTION

reentrant pure code which facilitates the multiple use of a single physical copy of any code section

2./ Global data and Linkage Section

contains the program or problem global variables and the necessary linkage data to interact with other

routines. It's split into two subsections

a./ Simple Variable Subsection which contains
- %OWN, %CONST integers, reals
- frame information and address constant used by the code section
- printers to space consuming values put in TABLE subsection.

a./ Table Subsection which contains
- %STRING constants
- %OWN arrays
- %SWITCH vestors
- External Reference and Entry Point information

3./ Stack Section
contains the data space required by the user's program and allocated in a dynamic fashion as the program proceeds.

External Reference and Entry Point description blocks for data and code reference points are chained into separate lists and their listheads are stored in the file header of the program file. At load the time they are substracted and queued up to the appropriate listheads in US notebook Service. PCF uses this information to satisfay the run time external references unresolved up to the point. Having made inquiries about the available service points it either reloads the descriptor block to point to the called routine or leaves it unchanged and services the remote connection through PCF actions every time a reference occurs.

A system with the above proposed structure can support a distributed way of working because

- it can provide the necessary multiplicity of resource components by pressing for an overall portability /of special programs, compilers, file system, support system etc./ The dynamic assigment of these resources is facilitated by the library system/ named services

- files/ and achieved through the run time satisfaction of the external references.
- the physical distribution of the system components and the utilisation of a communication network in case of interactions are paramount.
- the unity of system operation is guaranteed by the design philosophy and is based on self stabilizing algorithms. The application of "trial and error" policies using bidding algorithms and parallelism is inevitable because accurate system information is impossible to accumulate.
- the system is totally transparent, services can be called only by name and even the peculiarities of the conventional file services are well hidden behind the interprocess communication scheme.
- the constituent parts of the system have preserved their autonomy besides the cooperativity. The host processors mainly provide the most costeffective local service to their users and sell only the unused capacity of their own via the network. Systems would look for remote resources only in cases when their home ones heavily loaded or don't exist at all.

Conclusions

A computing machine, serving vast number of people through user definable interfaces should be high level programmable. Such computer with extreme reliability and accersibility features can be economically implemented as a distributed system.

All the technical details of the implementation method should be hidden from the user.

New inherently communicative supervisors are wanted, which can collectively support a universal, transparent and distributed, message based interprocess communication mechanism taking advantage of a special distributed firmware pripheral device, the /public/ data network.

Throughout the design OSI can serve not only as a reference language but as a conceptual design utility too.

References

1./ On distributed communication networks
P. Baran IEEE TRans. CS-12 /1969/

2./ Számitógép hálózatok. A magasszintü számitógép hálózatok tervezési kérdései.
Póka Péter KFKI /1978/

3./ Open Systems Interconnection
ISO/TC /SC 16 /1978/

4./ A system for interprocess communication in a resource sharing computer network.
D.C. Walden CACM VOL 15 NO 4 /1972/

5./ A high level framework for network based resource sharing
J.E. WHITE; Proc AFIPS 1976

6./ Description of the interactive supervisor of the system ANSWER
J. SOMOGYI Softtech D4-a /1977/

7./ What is a distributed data processing system?
P.J. Enslow Jr. Computer JAN 1978.

ОБ АВТОМАТИЗАЦИИ ПОСТРОЕНИЯ ЧИСЛЕННО-АНАЛИТИЧЕСКИХ МОДЕЛЕЙ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, ОПИСЫВАЕМЫХ МАРКОВСКИМИ ПРОЦЕССАМИ

Г.И.Праневичюс

Каунасский политехнический институт

(СССР)

В работе приводится методика автоматизации построения численно-аналитических моделей систем, описываемых марковскими процессами. Основное внимание сосредоточено на описании системы на языке событий и структуре транслятора генерирующего программы для автоматизированного построения уравнений Колмогорова. Приводится описание численно-аналитической модели концентратора сообщений сети ЭВМ.

I. ВВЕДЕНИЕ

При исследовании сложных вычислительных систем широко применяется имитационное моделирование, позволяющее определить характеристики системы. Однако в тех случаях, когда нужна высокая точность, имитационное моделирование требует много машинного времени. Аналитические исследования являются наиболее полными и исчерпывающими, но они позволяют получать результаты в замкнутом виде лишь для сравнительно узкого класса систем. В связи с этим важное значение приобретают численные методы исследования систем. В [I - 3] работах изложен подход численного анализа систем, описываемых марковскими процессами с дискретным множеством состояний и непрерывным временем. Предлагаемый подход построения численно-аналитических моделей включает четыре этапа: I) определение вектора состояний марковского процесса; 2) составление

уравнений, описывающих марковский процесс; 3) решение составленной системы уравнений; 4) вычисление вероятностных характеристик системы.

В данной работе будут рассмотрены некоторые вопросы автоматизации построения численно-аналитических моделей (ЧАМ) для нахождения средних характеристик системы, таких как средние длины очередей и обслуживания, коэффициентов простоев и т.п. Эти характеристики определяются из стационарных вероятностей системы, которые находятся из системы линейных уравнений, полученных из уравнений Колмогорова при $t \to \infty$. Следует отметить, что не все этапы построения ЧАМ в одинаковой степени поддаются автоматизации. Определение вектора состояний марковского процесса системы, которой ЧАМ создается, не поддается автоматизации. Вектор состояний системы определяется эвристически. Для правильного определения вектора состояний необходимо хорошее понимание функционирования системы. Основным требованием на этом этапе является введение достаточного числа координат, чтобы процесс, описывающий функционирование системы, был бы марковским.

Наиболее трудоемким этапом при создании ЧАМ является составление уравнений, которые описывают функционирование системы. Число уравнений обычно является большим (тысячи, десятки тысяч). В связи с этим очень важным вопросом является создание алгоритмов, которые позволяли бы автоматизированно строить нужные уравнения. В данной работе будет приведена методика описания функционирования системы на языке событий, а также структура транслятора, которая по описанию системы на языке событий создает программу для автоматизированного составления уравнений.

Известные численные методы решения линейных уравнений позволяют разрабатывать алгоритмы для расчета стационарных вероятностей на ЭВМ. Среди численных методов определения стационарных

вероятностей следует отметить метод последовательного вложения марковских цепей [I-3] и итерационный метод. Вышеуказанные методы расчета стационарных вероятностей особенно эффективны, когда они используются совместно с алгоритмами генерирования уравнений. Методы расчета стационарных вероятностей в данной работе не рассматриваются.

Только частично можно автоматизировать этап расчета характеристик моделируемой системы. Характеристики моделируемой системы рассчитываются по заранее составленным формулам, в которые входят параметры системы и стационарные вероятности системы.

2. ОПИСАНИЕ СИСТЕМЫ НА ЯЗЫКЕ СОБЫТИЙ

Пусть S и $\mathcal{D}$ соответственно морфологическое и функциональное описание системы [4]. M – методика, по которой создается функциональное описание системы $\mathcal{D}$, L – язык событий, на котором создается описание $\mathcal{D}$, T – программа-транслятор, которая переводит функциональное описание системы из формы на языке L на форму P, где P – описание системы на алгоритмическом языке программирования, напр. *FORTRAN* или *PL* – I, которая для заданного состояния системы генерирует все соседние ей состояния и устанавливает интенсивности переходов между заданным состоянием и соседними.

Вышесказанное можно записать в следующем виде:

$$S \xrightarrow{M} \mathcal{D} \xrightarrow{T} P.$$

При описании системы на языке событий L используются следующие данные:

I. Вектор состояний системы $N(t)=\{n_1(t), n_2(t), \ldots, n_{KSK}(t)\}$, где KSK – число координат вектора состояний. Координаты вектора состояний являются дискретными величинами и определяют

состояния отдельных элементов системы в момент времени t . Напр.,длины очередей, занято устройство обслуживания или нет, из какого потока обслуживается заявка и т.д.

2. Множество событий системы $E=\{E_1, E_2, \ldots, E_M\}$, которые могут произойти в системе в дискретные моменты времени. Примерами событий могут быть: приход заявки на обслуживание, завершение обслуживания и т.д.

3. Множество интенсивностей переходов $\lambda=\{\lambda_1, \lambda_2, \ldots, \lambda_M\}$.

4. $KM=2\times M$ - максимальное число соседних состояний. Соседним состоянием для данного состояния будем называть такое, в которое система может перейти из данного состояния, если произошло одно из событий из множества E , или такое, из которого система может перейти в данное состояние, если произошло одно из событий множества E .

$P1$ - интенсивность перехода из данного состояния в соседнее,

$P2$ - интенсивность перехода из соседнего состояния в данное.

При описании системы на языке событий используются операторы: V - название системы. Этот оператор указывает начало описания системы;, P - вспомогательное выражение описания событий, этот оператор используется, если надо ввести обозначения, выражения, сокращения или формулы перед описанием всех событий или внутри описания события; S - условие события. С помощью этого оператора указывается начало описания события и описывается условие (условия), при котором (которых) это событие может произойти, чтобы система перешла из соседнего состояния в данное, или из данного состояния в соседнее; K - оператор изменения координат состояния. Этот оператор всегда должен находиться внутри описания события. С помощью этого оператора ука-

зывается, какие и как должны изменяться координаты; *KS* – условие изменения координат состояния. Этот оператор используется только перед оператором *K* и только с этим оператором. С помощью оператора *KS* описывается условие (условия), при котором (которых) должны изменяться координаты, указанные в следующих после *KS* операторах *K* ; *I* – оператор определения интенсивностей переходов между состояниями. Этот оператор используется в конце описания каждого события. С помощью оператора *I* определяются интенсивности переходов *P1* и *P2* ; *IS* – условие установки интенсивностей переходов между состояниями. Этот оператор используется только перед оператором *I* и только с этим оператором. С помощью оператора *IS* описывается условие (условия), при котором (которых) интенсивности переходов устанавливаются в следующих после *IS* операторах *I* ; *E* – конец описания всех событий.

Так как в множество соседних состояний входят состояния, в которые из данного состояния можно попасть и состояния, из которых можно в данное состояние попасть, то каждое событие из множества *E* надо описать дважды:

– описание $\mathfrak{D}'$ описывает переходы в те состояния, в которые можно попасть из данного состояния за один шаг;

– описание $\mathfrak{D}''$ описывает переходы из тех состояний, из которых можно попасть за один шаг в данное состояние.

Возможная последовательность операторов для описания одного события:

$$S[(P]\cdots][KS]K[(K]\cdots][[KS \lrcorner K][(K]\cdots]]\cdots]I[$$
$$[[IS \lrcorner I][[[I]\cdots]]\cdots] ,$$

где [ ] означает, что содержание между фигурными скобками может быть опущено; ... означает, что содержание в скобках

перед ··· может быть повторено несколько раз.

Возможная последовательность операторов описания системы:

$$V([P]\cdots]AA[[A]\cdots]E ,$$

где A означает описание одного события.

Ниже приводится алгоритм, по которому составляется описание на языке событий.

I. $i := 1$.

2. Описание условий для координат вектора состояний, чтобы было возможно событие E_i, вследствие которого система переходит из данного состояния в соседнее (оператор S).

3. Нужна ли дополнительная проверка условий для изменения координат состояния при событии E_i? Если нет - перейти к п. 5.

4. Описание дополнительных условий для изменения координат для данного состояния (оператор KS).

5. Описание изменений координат системы при событии E_i (оператор K).

6. Описаны ли все дополнительные условия для изменения координат состояния при событии E_i? Если нет - перейти к п. 3.

7. Нужны ли дополнительные условия для определения интенсивностей переходов $P1$ и $P2$ при событии E_i? Если нет - переход к п. 9.

8. Описание дополнитедьных условий для интенсивности перехода при событии E_i (оператор IS).

9. Определение интенсивностей переходов $P1$ и $P2$ при событии E_i (оператор I).

IO. Описаны ли все дополнительные условия для определения интенсивностей переходов $P1$ и $P2$ при событии E_i? Если нет - переход к п. 7.

II. $i = M$? Если да - переход к п. I6.

I2. $i := i+1$.

I3. Это первый просмотр множества событий E ? Если да - переход к п. 2, если нет - к п. I5.

I4. $i := 1$.

I5. Описание условий для изменения координат вектора состояний, чтобы было возможно событие E_i , по которому система перешла бы из соседнего состояния в данное (оператор S). Переход к п. 3.

I6. Кончился ли второй просмотр множества событий E ? Если нет - переход к п. I4.

I7. Конец описания событий.

3. ТРАНСЛЯТОР ГЕНЕРИРОВАНИЯ ПОДПРОГРАММ ДЛЯ ОПРЕДЕЛЕНИЯ СОСЕДНИХ СОСТОЯНИЙ

Исходными данными для транслятора, генерирующего подпрограмму для определения соседних состояний, является описание системы на языке событий по вышеприведенной методике.

Исходными данными для подпрограммы определения соседних состояний является состояние x и номер соседнего состояния k, $k = \overline{1, KM}$.

Это можно записать в виде функционала

$$F(x, k) = (y, P1, P2),$$

где y - соседнее x-cy состояние;

$P1$ - интенсивность перехода из состояния x в y ;

$P2$ - интенсивность перехода из y в x .

Если $P1$ или $P2$ равен нулю, это означает, что соответствующего перехода нет.

Алгоритм действий транслятора состоит из двух основных этапов, которые повторяются для каждого входного оператора.

I. Ввод и распознование оператора.

2. Обработка оператора: генерирования последовательности операторов Фортрана по типу и содержанию оператора.

4. СТРУКТУРА ПАКЕТА ПРОГРАММ АВТОМАТИЗИРОВАННОГО ПОСТРОЕНИЯ ЧИСЛЕННО-АНАЛИТИЧЕСКИХ МОДЕЛЕЙ

На рис. I структура пакета программ автоматизированного построения численно-аналитических моделей систем, описываемых марковскими процессами [5].

В пакет программ входит: транслятор, который по описанию функционирования системы на языке событий генерирует подпрограмму для генерирования линейных уравнений системы; программа контроля описания, которая проверяет корректность описания системы на языке событий; программа расчета стационарных вероятностей методом последовательного вложения марковских цепей; программа расчета стационарных вероятностей итерационным методом; программа расчета характеристик системы; управляющая программа, которая устанавливает желаемый режим счета; библиотека подпрограмм генерирования систем линейных уравнений.

Состояния марковских процессов (МП) задаются целыми, неотрицательными числами. Так как реальные системы описываются многомерными МП, то для перехода от многомерного МП к одномерному используются формулы для преобразования вектора N в число X и обратно:

$$X=\sum_{i=1}^{\rho} n_i \cdot \pi_i \quad , \quad \pi_i=\prod_{j=i+1}^{\rho}(m_j+1), \quad i=\overline{1,\rho} \, ,$$

где ρ – размерность МП;

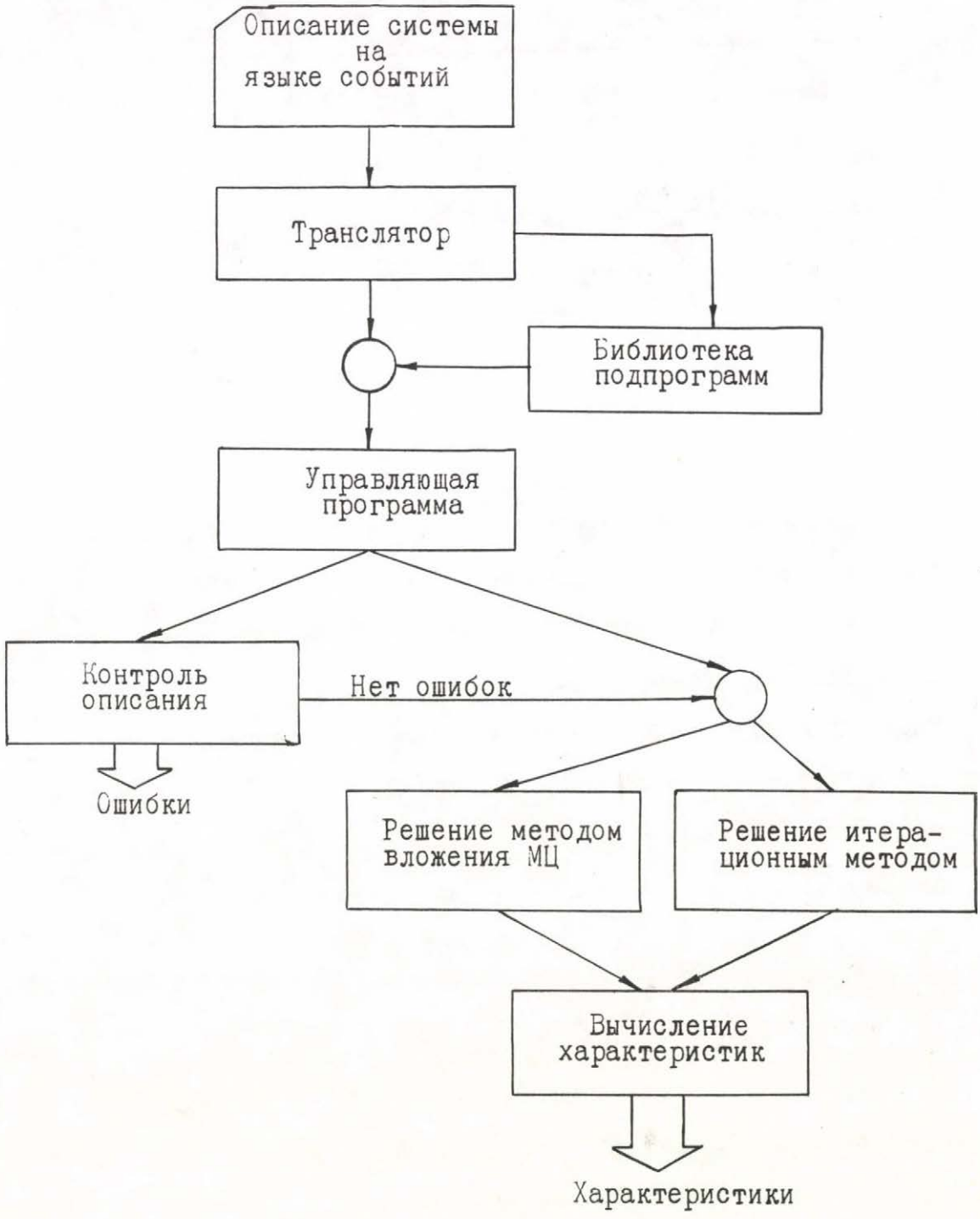


Рис. I. Структура пакета программ.

$\overline{N}=(n_1, n_2, \ldots, n_p)$ – состояние МП, $n_i \leqslant m_i$, $i=\overline{1,p}$;

X – целое неотрицательное число.

Обратное преобразование осуществляется по формулам:

$$r_0 = X,$$
$$n_i = \left\lfloor \frac{r_{i-1}}{\pi_i} \right\rfloor, \quad i=\overline{1,p},$$
$$r_i = r_{i-1} - n_i \cdot \pi_i .$$

5. ДИСТАНЦИОННЫЙ КОНЦЕНТРАТОР СООБЩЕНИЙ СЕТИ ЭВМ

Морфологическое описание системы. Функционирование дистанционного концентратора сообщений (ДК) представляется следующей системой массового обслуживания (рис. 2). Считается, что потоки сообщений, поступающие в ДК, являются пуассоновскими, а законы распределения времен обслуживания – экспоненциальные. На ДК поступает два потока сообщений с интенсивностями λ_1 и λ_2. Поток с интенсивностью λ_1 поступает от абонентских пунктов, поток с интенсивностью λ_2 – из магистрального канала передачи данных. Заявки, поступившие из абонентских пунктов, поступают в очередь Q_1, которая ограничена величиной K_1. Пришедшая заявка и заставшая в очереди K_1 заявок получает отказ. Заявки, поступившие из магистрального канала, поступают в очередь Q_2, которая ограничена величиной K_2. Заявки из обеих очередей обслуживаются одним процессором с интенсивностями μ_1 и μ_2 соответственно. Заявки из очереди Q_2 имеют абсолютный приоритет перед заявками из очереди Q_1. Заявки из очередей Q_1 и Q_2 обслуживаются только тогда, когда суммарная длина очередей Q_1 и Q_2 меньше заданной величины m, т.е. когда в оперативной памяти ДК есть свободное место. Заявки из очереди Q_1 после обслуживания поступают в очередь Q_3, откуда выводятся из ДК с интенсивностью μ_3. Аналогично

заявки из очереди Q_2 поступают в очередь Q_4, откуда выводятся с интенсивностью μ_4.

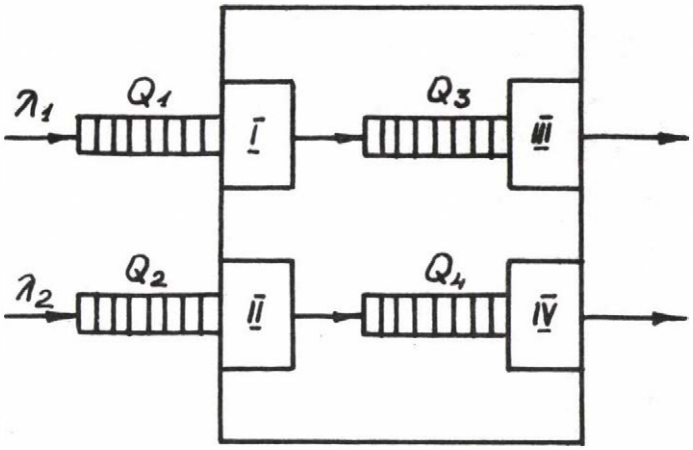


Рис. 2. Структурная схема дистанционного концентратора сообщений

Состояние ДК описывается следующим многомерным вектором:

$$X(t)=\{n_1(t), x_1(t), n_2(t), x_2(t), n_3(t), n_4(t)\},$$

где $n_i(t)$, $i=\overline{1,4}$ – длины очередей Q_i, $i=\overline{1,4}$;

$$x_1(t)=\begin{cases} 1, & \text{если вводится сообщение, поступившее} \\ & \text{из абонентских пунктов в ДК;} \\ 0, & \text{в противном случае;} \end{cases}$$

$$x_2(t)=\begin{cases} 1, & \text{если вводится сообщение, поступившее} \\ & \text{из магистрального канала;} \\ 0, & \text{в противном случае.} \end{cases}$$

Число координат вектора состояний системы: KSK = 6.

Ограничения на длины очередей:

$$n_1 \leqslant k_1,$$
$$n_2 \leqslant k_2,$$
$$n_3+n_4 \leqslant m.$$

Множество событий, которые могут произойти в ДК-е состоят

из шести событий. E_1 , E_2 - приход заявок в очереди Q_1 и Q_2 соответственно, E_3 , E_4 , E_5 , E_6 -завершение обслуживания заявок в очередях Q_1 , Q_2 , Q_3 , Q_4 соответственно.

Массив интенсивностей переходов между состояниями системы:

$$INTENS = (\lambda_1, \lambda_2, \mu_1, \mu_2, \mu_3, \mu_4).$$

Массив ограничений координат вектора состояний системы:

$$MAX = (m_1 = k_1, m_2 = 1, m_3 = k_2, m_4 = 1, n_3 = m, n_4 = m).$$

Описание системы на языке событий.

Описание $\mathfrak{D}'$:

E_1: S $n_1 < k_1$
K $n_1 = n_1 + 1$
KS $x_1 = 0 \wedge n_3 + n_4 < m \wedge n_2 = 0$
K $x_1 = x_1 + 1$
I $P1 = \lambda_1$, $P2 = 0$.

E_2 : S $n_2 < k_2$
K $n_2 = n_2 + 1$
KS $x_2 = 0 \wedge n_3 + n_4 < m$
K $x_2 = x_2 + 1$
KS $x_1 > 0$
K $x_1 = x_1 - 1$
I $P1 = \lambda_2$, $P2 = 0$.

E_3: S $x_1 > 0$
K $n_1 = n_1 - 1$, $n_3 = n_3 + 1$
KS $n_1 = 1 \vee n_3 + n_4 + 1 = m$
K $x_1 = x_1 - 1$
I $P1 = \mu_1$, $P2 = 0$.

E_4: S $x_2 > 0$
K $n_2 = n_2 - 1$, $n_4 = n_4 + 1$
KS $n_2 = 1 \vee n_3 + n_4 + 1 = m$
K $x_2 = x_2 - 1$
KS $n_1 > 0 \wedge n_3 + n_4 + 1 < m \wedge n_2 = 1$
K $x_1 = x_1 + 1$
I $P1 = \mu_2$, $P2 = 0$.

E_5: S $n_3 > 0$
K $n_3 = n_3 - 1$
KS $x_2 = 0 \wedge n_2 > 0$
K $x_2 = x_2 + 1$
KS $x_1 = 0 \wedge n_2 = 0 \wedge n_1 > 0$
K $x_1 = x_1 + 1$
I $P1 = \mu_3$, $P2 = 0$.

E_6: S $n_4 > 0$
K $n_4 = n_4 - 1$
KS $x_2 = 0 \wedge n_2 > 0$
K $x_2 = x_2 + 1$
KS $x_1 = 0 \wedge n_2 = 0 \wedge n_1 > 0$
K $x_1 = x_1 + 1$
I $P1 = \mu_4$, $P2 = 0$.

Списание $\mathfrak{D}''$:

E_1: S $n_1>0$
K $n_1=n_1-1$
KS $n_1=1 \wedge x_1>0$
K $x_1=x_1-1$
I $P1=0$, $P2=\lambda_1$.

E_2: S $n_2>0$
K $n_2=n_2-1$
KS $n_1=1 \wedge x_1>0$
K $x_1=x_1-1$
KS $n_2=1 \wedge x_2>0 \wedge n_1>0$
K $x_1=x_1+1$
I $P1=0$, $P2=\lambda_2$.

E_3: S $n_2=0 \wedge n_3>0 \wedge n_1<K_1$
K $n_1=n_1+1$, $n_3=n_3-1$
KS $x_1=0$
K $x_1=x_1+1$
I $P1=0$, $P2=\mu_1$.

E_4: S $n_4>0 \wedge n_2<K_2$
K $n_2=n_2+1$, $n_4=n_4-1$
KS $x_2=0$
K $x_2=x_2+1$
KS $x_1>0 \wedge x_2=0$
K $x_1=x_1-1$
I $P1=0$, $P2=\mu_2$.

E_5: S $n_3+n_4<m$
K $n_3=n_3+1$
KS $n_2>0 \wedge n_3+n_4+1=m \wedge x_2=1$
K $x_2=x_2-1$
KS $n_1>0 \wedge n_3+n_4+1=m \wedge x_1=1 \wedge n_2=0$
K $x_1=x_1-1$
I $P1=0$, $P2=\mu_3$.

E_6: S $n_3+n_4<m$
K $n_4=n_4+1$
KS $n_2>0 \wedge n_3+n_4+1=m \wedge x_2=1$
K $x_2=x_2-1$
KS $n_1>0 \wedge n_3+n_4+1=m \wedge x_1=1 \wedge n_2=0$
K $x_1=x_1-1$
I $P1=0$, $P2=\mu_4$.

Граф состояний ДК для случая, когда $K_1=K_2=m=1$ представлен на рис. 3. На рис. 4 приведены зависимости средних длин очередей $M(n_i)$, $i=\overline{1,4}$ от ϱ_1 и ϱ_2, где $\varrho_1=\frac{\lambda_1}{\mu_1+\mu_3}$, $\varrho_2=\frac{\lambda_2}{\mu_2+\mu_4}$. Расчеты проводились при $K_1=K_2=2$, $m=4$.

6. ЗАКЛЮЧЕНИЕ

Изложенный в данной работе метод автоматизированного построения численно-аналитических моделей и расчета характеристик системы

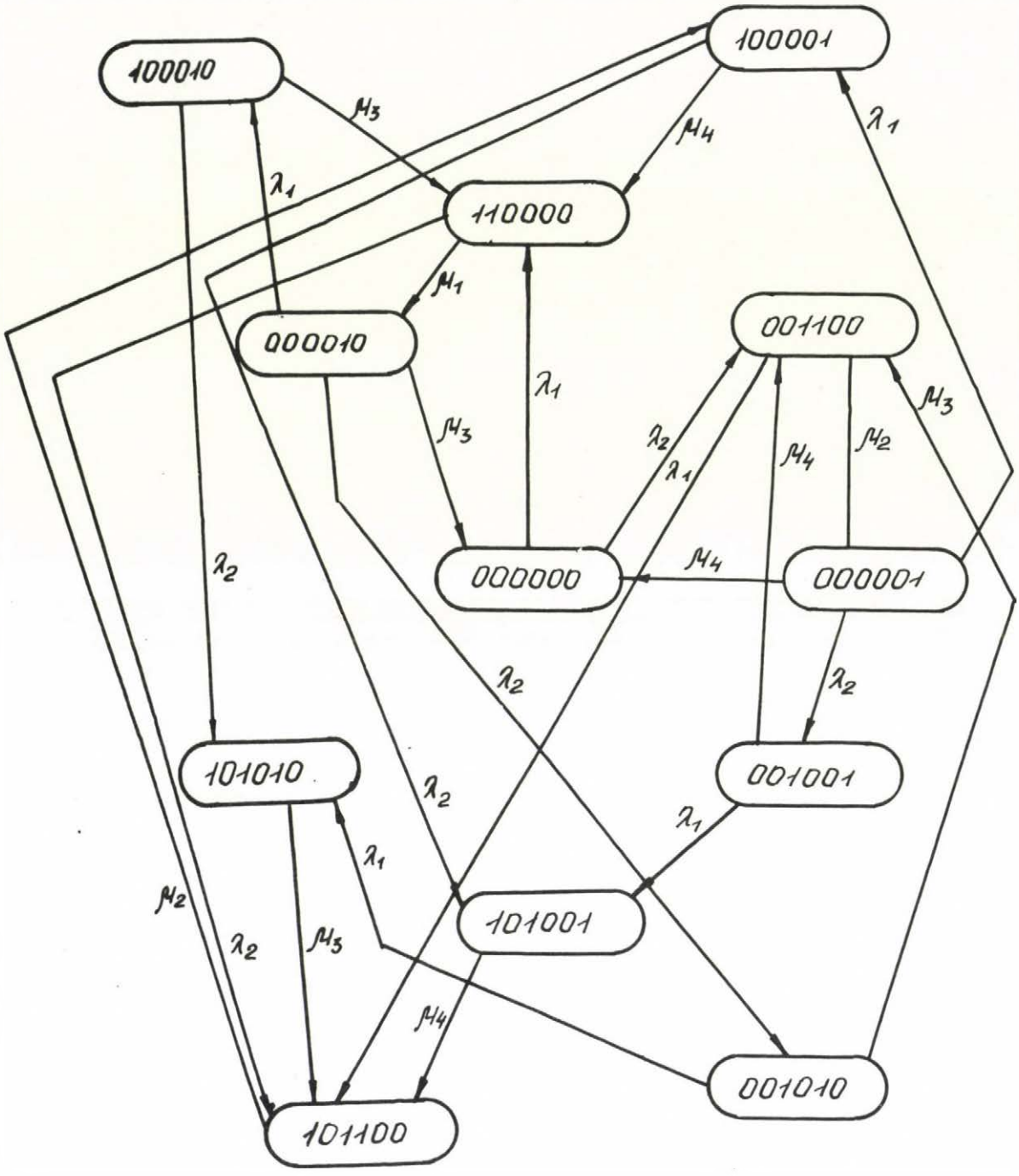


Рис.3. Граф состояний ДК-а.

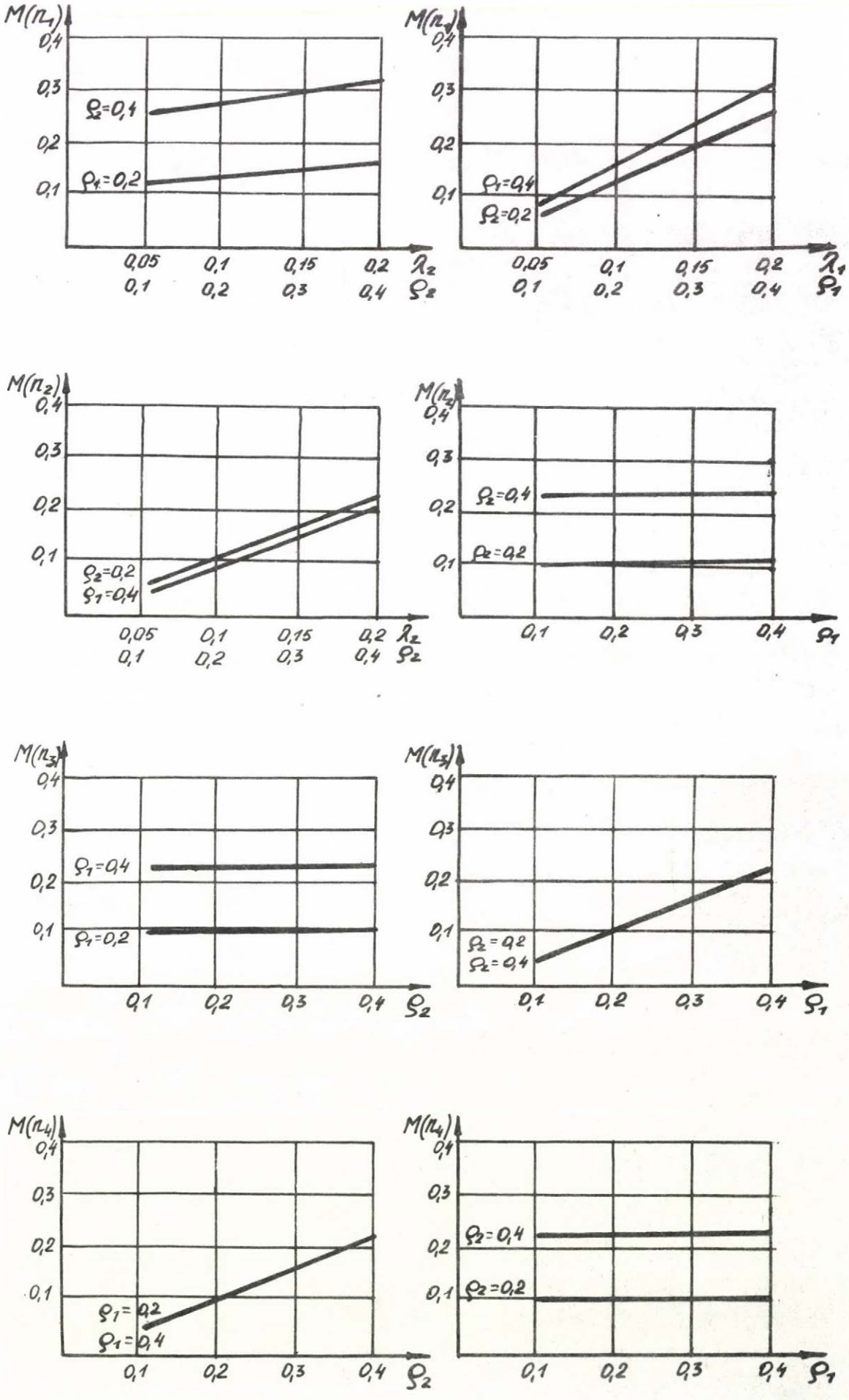


Рис. 4. Зависимости длин средних очередей.

не зависит от структуры и алгоритмов функционирования системы, так как функционирование системы задается её описанием на языке событий. Единственным ограничением предлагаемого метода является размерность уравнений. Предлагаемый в работе метод позволяет значительно сократить время построения численно-аналитических моделей.

ЛИТЕРАТУРА

I. А.Вайткявичюс, З.Навицкас, Г.Праневичюс. Вычисление стационарного распределения эргодических марковских цепей на ЭЦВМ. Лит. мат. сб., т. XУI, № I, I976, с. 2I7-220.

2. А.Вайткявичюс, Г.Праневичюс. Вычисление стационарного распределения марковской цепи, заданной в бесконечном фазовом пространстве. Математика и математическое моделирование. Вып. I, Каунас, I976, с. I9-23.

3. Г.И.Праневичюс. Численно-аналитическое моделирование систем, описываемых марковскими процессами. Теория сложных систем и методы моделирования. Труды семинара. М., I979, с. 52-64.

4. В.В.Дружинин, Д.С.Конторов. Проблемы системологии . - М., Советское радио, I976, 296 с.

5. А.Вайткявичюс, Г.Праневичюс. Структура пакета прикладных программ численно-аналитического моделирования. Математика и математическое моделирование. Вып. 2, Вильнюс, I977, с. 37-40.

Notes on Portability of Operating Systems

J. Somogyi
Research Institute for Applied Computer Sciences,
Budapest

1. Introduction

The input-output system of contemporary computers seems - at least at first sight - like a jungle. Many different input-output systems have been conceived and implemented. A detailed examination shows, however, that apart from extreme cases /e.g. machines without interrupt system/ the overwhelming majority is classifiable into three categories.

1.1. Input-output processors

In most of the modern digital computers the input-output operations are performed by autonomous channels. The channels are dedicated processors, specially designed for handling input-output operations. When the central processor executes an input-output instruction, it initializes the channel and turns the control of the input-output operation over to the channel rather than carrying out the input-output operation itself.

Just like central processors, the channels are capable of executing instructions, which are called channel commands. A sequence of such commands forms a channel program which allows many versatile input-output operations concurrent with the operation of the central processor.

1.2. Single instruction devices

The main characteristic of these systems is, that the peripheral device executes one input- output operation /corresponding to one channel command/ at a time. The devices have memory address, byte count, command, status, etc. registers, and the devices are controlled by the contents of these registers.

1.3. Direct program control

In case of direct program control, one character is transferred by the execution of one input-output instruction. To transfer a complete block of data, the execution of a number of instructions is required. Direct program control is the simplest organization for input-output control. It was used for early computers, but it is still employed for some inexpensive computers of today.

Since direct program control confronts the operating system with an unsolvable task in the case of fast devices, direct program control appears together with a single instruction system in real installations.

2. Portability problems

When an operating system is to be moved from a machine equipped with input-output processors to another machine, which has single instruction devices, the channel programs must be converted to central processor programs. This is not a porting procedure in the usual sense. Special problems are coming from both the hardware and software conditions.

2.1. Hardware problems

The programs running on machines equipped with input-output processors consist of several concurrently executable parts. A reasonable program contains at least three parts: the input program, the processing program and the output program. Two of these, the input and output programs are executed by channels, while the processing program is executed by the central processor. The communication between the parts links them together to form a complete program. /E.g. the processing program tells the input-output programs the address and length of the input-output area./

This is a typical case of parallel processing. In this situation there are two problems to be solved: the synchronization and the communication of the processes.

As to the synchronization the hardware provides satisfactory solution. The channels can be activated by the central processor by making use of the input-output instructions, while the channels can signal the end of the channel programs by means of interrupts.

As regards the communication, there are serious problems due to the lack of addressing modes. The channel commands have two operands: the memory address and the byte count, of which the memory address must be an absolute address, while the byte count must be constant. Therefore the input to the channel program must be coded into the channel commands statically, or the channel program has to be modified dynamically. Portability means, that such channel programs would be converted into central processor programs for another machine. This is illusory.

Moreover, one may think of the direct program control as portable. This is not the case. Apart from efficiency

considerations /which may not be neglected in such a large scale/, the direct program control works only on direct program controlled hardware.

2.2. Software problems

While the hardware problems make the portability of the channel programs impossible, the software problems prevent the writing of that programs.

The reasons are mainly historical. The existence of the channel programs, and therefore the parallel nature of the programs are hidden from the high level language programmers. Still on machines having no channels the input-output operations are executed by the operating system, i.e. outside of the program. Therefore high level languages /not intended for writing operating systems/ treat the input-output operations badly.

Consequently, using the present-day high level languages, one has no formal means to write channel programs. But this is the lesser evil. In addition, the channel programs have to be marked explicitly as parallel branches whithin the program. That is, concurrent language has to be used.

3. A possible way out

According to the input-output system classes established above, we define the following input-output levels.

3.1. Program level

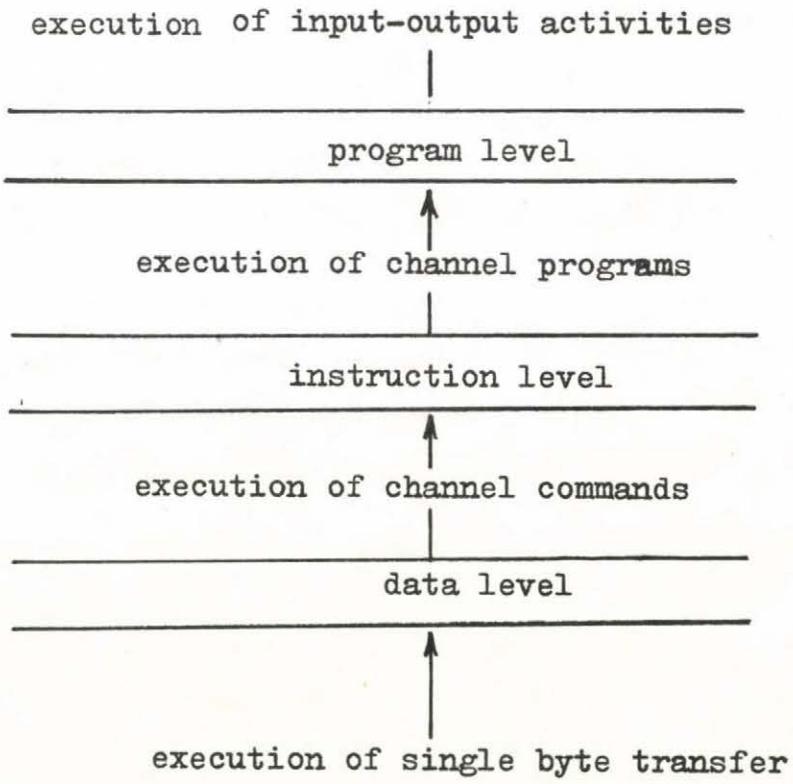
At this level the operating system assumes the existence of channels, which are capable of executing complete channel programs. Moreover, it is assumed that there are central processor instructions by which the channels can be activated, and that the end of the channel programs is signalled by interrupts.

3.2. Instruction level

At this level the operating system assumes that the peripheral devices are able to execute one input-output operation /read, write, etc./. Moreover it is assumed that there are memory address, byte count, command, etc. registers, by which the devices can be controlled.

3.3. Data level

At this level the operating system assumes that the devices transfer one character in response to one input-output instruction.



The levels defined in this manner have the common characteristic that they can be placed on existing hardwares. But the transfer from one machine to another is not porting in the usual sense. Passing over to a higher level machine is accomplished by deleting all the lower levels from the system. Change over from a higher level machine to a lower level one means the inclusion of the required levels into the system.

Moving between machines at the same level is the porting in the usual sense.

It has to be noted that although on machines having no channels the channel programs are interpreted by the operating system, this interpretation has no concern with efficiency. In every operating system the programs present their input-output requirements to the system in some form, and the system interprets that form. In the proposed solution this form is the channel program. What is more, this form allows to present multiple requests in one system call.

This does not mean, that the problem of writing channel programs is shifted upon the users. Programs using the file system of the operating system do not even need to know about channel programs.

Specifications of reliable software

L. Varga

Abstract

In this paper the type abstraction is discussed as a useful methodological tool in the development of reliable programs. It begins with an overview of specification techniques. The formal specification methods for abstract data types are then discussed. The Hoare's axiomatic approach and the Guttag's algebraic approach to type abstraction are examined. It is shown that the verification method for axiomatic specification can be used for verifying the correctness of an implementation against algebraic specification.

1. Introduction

In the last decade, software developers have attempted to develop larger and larger programming systems. A considerable percentage of these systems were unseccessful. The most important reason for the unseccesfulness of these efforts was the great psychological complexity of understanding these systems. This is why the key problem in the development of large programs is to reduce the amount of complexity that must work with at any one time.

A useful methodological tool for accomplishing this is the use of abstraction.

Programs can be decomposed into seperate components which can be specified by means of abstractions, seperating essential properties from inessential details. Similarly, the complexity of attributes associated with a data object can be reduced by separating those properties that are relevant to programs using the data object from the details of its implementation. This idea was introduced by Hoare in [4] specifying abstract data types as a programming tool. This methodological tool is discussed in this paper.

The next section gives an overview of specification methods. Section 3 then briefly describes the different approaches to the formal specification of abstract data types. In Section 4 the Hoare's axiomatic approach to type abstraction is presented. The main concern of this section is to describe a techniques for verifying the correctness of implementations for axiomatic specifications.
In Section 5 this verification technique is used for verifying the correctness of implementations for algebraic specifications of abstract data types.

2. An overview of specification methods

There are many approaches to specifications, among them both formal and informal approaches take place.

Informal specifications. Among the informal specifications the traditional form is the natural-language

specification. The ambiguity of this specification is well known, which leads to serious problemes in the inter-relationships between the specification and other phases of the software life cycle. Therefore different kinds of formatted specification languages were developed. These languages express the specification in a standardized syntax with an imprecise semantics. A good deal of progress has been made in the last few years in the development of programming system supporting formatted specifications, for example ISDOS/CADSAT [10], SAMM [5] and AXES [6].

The verification view of specifications reguires a greater degree of formalism than is present in informal specification.

Formal specifications, which have a precise mathematical form, with both syntax and semantics precisely defined. A good summary of these specifications can be found in [9]. The formal specification methods are begining to be used in developing practical software products. The main result of this progress are the formal specification of abstract data types and the incorporation of abstract data types into many new programming languages, e.g., Clu [8], Alphard [11], Euclid [7].

3. Specification of abstract data types

There are two different approaches to the formal specification of abstract data types. One is the procedural approach and another is the non procedural one.

In a procedural approach, a set of procedures is given by programs which correspond to meaningful operations on the instances of the abstract type. The "class" mechanism of Simula 67 [1] can be used for the incorporation of abstract types in Simula 67 using a procedural approach. This types declaration has the form:

```
class T;
    begin declarations of  c1, c2,...,cn;
        procedure  p1  <formal parameter part> ; Q1;
        procedure  p2  <formal parameter part> ; Q2;
                :
        procedure  pm  <formal parameter part> ; Qm;
        Q
    end;
```

where $p_1,\ldots,p_m$ are procedure names with $Q_1,\ldots,Q_m$ procedure bodies respectively, and Q is a program which assigns initial values to the variables $c_1,\ldots,c_n$.

Having declared this representation for the type T in an abstract program we can declare a variable s of type T :

```
var (T) s ;
```

and can use the operations upon it:

$s.p_i$ < actual parameter part > $\quad i=1,2,\ldots,m.$

In a procedural abstraction, the level of abstraction achived by the specification depends upon the level of abstraction of the programming language which is used. For

example, a language like Vienna Definition Language allows more abstract specification than the language Simula 67.

The procedural specification is very similar to programming and therefore it can be relatively easily used by software designers. However this specification bears many disadvantages too. Most significantly, this is a "how" specification and in the case of a complex operation its unnecessary detail does not serve sufficiently the communication among the various groups of people who must use it.

In a non procedural specification we describe the properties of the abstract type relevant to programs using the abstraction. There are many variants of formalisms that can be used to create non procedural specifications. Among them, the Hoare's axiomatic specification [4] has been used in practice. An entirely different approach to specifying abstract types is the algebraic specification [2]. We shall be dealing with these two specification techniques in the remainder of this paper.

4. Hoare's approach to the specification of abstract types

An axiomatic specification of abstract type provides separate pictures to its user and its implementor. To the user the specification defines a set of abstract objects and a set of operations associated with the type.

The set of abstract objects

$$\{ a \mid I_a(a) \}$$

is given by the predicate $I_a(a)$ which is called abstract invariant.

The abstract operations are defined by their pre - and post-conditions:

$$\{ pre_a (a') \}\ a = f_a(a')\ \{ post_a (a) \}$$

where pre_a, $post_a$ are predicates.

The initial abstract object, from which every instance of the abstract objects can be generated through application of the abstract operations is given by a predicate $P_o(a)$.

To the implementor the specification presents representation and implementation information.

A representation description provides a set of concrete objects, a concrete initial object and a function which maps a concrete object to an abstract object.

A set of concrete objects is given in term of concrete invariant $I_c(c)$:

$$\{c \ / \ I_c(c)\}.$$

The mapping function

$$a = \varphi(c)$$

maps a concrete representation c to an abstract object a.

The implementation discription is a set of pre- and post- condition pairs:

$$\{pre_c(c')\} \quad c = f_c(c') \quad \{post_c(c)\}$$

A well known example is the abstract stack, which can be specified as follows:

Abstract invariant:

$$I_a(a) : \text{sequence}(a) \wedge 0 \leq \text{length}(a) \triangleq n \wedge n \geq 1$$

Initial abstract object a_o is given by the predicate

$$P_o(a_o) : \text{nullsequence}(a_o)$$

Abstract operations:

$\{0 \leq \text{length}(s') < n\}$ s = push(s',x) $\{s = s' \frown x\}$

$\{0 < \text{length}(s') \leq n\}$ s = pop(s') {s = leader (s')}

$\{0 < \text{length}(s') \leq n\}$ x = top(s') {x = last(s')}

Representation:

A concrete object c is represented by two variables v, sp where

$$\text{vector}(v) \wedge \text{integer}(sp)$$

Concrete initial object is given by sp=0.

Mapping function

$$a = \text{sequence}(v,1,sp)$$

/where

$$\text{sequence } (V,i,j) = V_{i+1}, V_{i+2}, \ldots, V_{i+j} \ )$$

Concrete invariant:

$$I_c(c) : \ 0 \leq sp \leq n$$

Implementation:

$\{0 \leq sp' < n\}$ push $\{sp = sp'+1 \wedge v[sp] = v\}$

$\{0 < sp' \leq n\}$ pop $\{sp = sp'-1\}$

$\{0 < sp' \leq n\}$ top $\{x = v[sp']\}$

An implementation can be verified by the following proof rule follows closely [11].

1. Show that the representation is correct. Given the set of the concrete objects

$$\{c \ / \ I_c(c)\}$$

and the initial object c_o, prove the following:

a./ if $c \in \{c \mid I_c(c)\}$ then $\varphi(c) \in \{a \mid I_a(a)\}$,

b./ $c_o \in \{c \mid I_c(c)\}$ and $P(\varphi(c_o)) \equiv$ true.

In our example, we have to show

$$0 \leq sp \leq n \supset 0 \leq \text{length } (v,1,sp) \leq n$$

and

$$0 \leq 0 \leq n \wedge \text{nullsequence } (\text{sequence } (v,1,0))$$

which are trivial.

2. Verify that the implementation is correct. Given the set of theorems

$$\{pre_c(c')\}\ \ c = f_c(c')\ \{post_c(c)\}$$

show the following:

a./ for each f_c

$$f_c(c) \in \{c \mid I(c)\},$$

i.e. show

$$\{pre_c(c') \wedge I_c(c')\}\ \ c = f_a(c')\{post_c(c) \wedge I_c(c)\}$$

b./ if an operation $f_a(\varphi(c))$ is legal then the operation $f_c(c)$ is legal too, i.e. show

$$pre_a\ (\varphi(c)) \wedge I_c(c) \supset pre_c(c)$$

c./ if the result c of a legal concrete operation statisfies the concrete postcondition $post_c$ then $\varphi(c)$ statisfies the abstract postcondition $post_a$, i.e. show

$$pre_c(c') \wedge I_c(c') \wedge post_c(c) \wedge I_c(c) \supset post_a\ (\varphi(c))$$

In our example, looking at the operation pop we have to verify the following theorems:

a./ $\{0 < sp' \leq n\}\ (v,sp) = pop(v,sp')\ \{0 \leq sp \leq n \wedge sp=sp'-1\}$

b./ $(0 < \text{length}(\text{sequence}(v,1,sp')) \leq n \wedge 0 \leq sp \leq n \supset$

$0 < sp \leq n$

c./ $0 < sp' \leq n \wedge 0 \leq sp' \leq n \wedge sp = sp'-1 \wedge 0 \leq sp' \leq n \supset$

$\text{sequence}(v,1,sp) = \text{leader}(\text{sequence}(v,1,sp'))$

5. Algebraic specification

An algebraic specification consists of two parts:

- the syntactic specification provides a set of function names together with domain and range definition;
- the semantic specification provides a set of axioms, which formally describes the properties of the functions and their relationships to one another.

The set of instances of an abstract type and the functions /operations/ associated with the type can be regarded as an abstract algebra.

Recently, many variants of algebraic specifications have been proposed. A good review can be found in [9]. Today, variants of Zilles' method predominate, e.g. [12], [2]. Our purpose now is to describe an algebraic specification method based upon Guttag's work [3] and suggest a way of verifying the correctness of an implementation according to an algebraic specification using the verification rules discussed above.

The algebraic specification method can be illustrated by a simple example:

Syntactic specification

Constructor functions

null : → structure

assign : structure x parameter x elem → structure

Selector functions

read : structure x parameter → elem

delete : structure x parameter → structure

Semantic definition

read (null,p) = error

read (assign(s,p,e),p') = if p = p' then e
else read(s,p')

delete (null,p) = null

delete (assign(s,p,e),p') = if p = p' then
delete (s,p) else assign (delete(s,p'),p,e)

Here we distinguished constructor functions from selector functions. The constructor functions that can be used to build every instance of this abstract type. A nonconstructor function is called selector functions.

The function 'null' creates a new instance of the abstract type and every instances of the type can be generated by application of the constructor function assign(s,p,e) one after the other.

The algebraic specification seems very simple, which defines only essential characteristics of a type including a large class of implementations.

The procedural nature of the semantics with which the

implementor must deal, can be reflected by an axiomatic specification rather than an algebraic specification. Therefore an axiomatic approach is more appropriate to describe an implementation.

A possible representation and implementation of our abstract type may be the following for the case of parameter = {1,2,...,n}.

Representation:

A concrete object c is represented by an array A and an integer k :

A : array [0...n] of elem

k : 0 ... n;

Concrete invariant:

$I_c(c)$: $0 \leq k \leq n$

Implementation:

{true} c = null {k=0 $\wedge$ ($\forall$i, 1 $\leq$ i $\leq$ n) (A[i] = 0)}

{$0 \leq k' \leq n \wedge 0 \leq i \leq n \wedge x \neq 0$}

(A,k) = assign ((A,k'),i, x)

{A = α ((A,k'),i,x) $\wedge$ (if A'[i] = 0 then k=k'+1 else k=k')}

{$0 < k \leq n \wedge 0 \leq i \leq n \wedge A[i] \neq 0$} e=read((A,k),i) {e=A[i]}

{$0 < k' \leq n \wedge 0 \leq i \leq n \wedge A'[i] \neq 0$} (A,k)=delete((A,k'), i)

(A = α((A,k'), i, 0) $\wedge$ k = k'-1}

where the expression $\alpha((A,k),i,x)$ means: the array A with the ith element replaced by x.

The mapping function $s = \varphi(A,k)$ can be defined as follows:

$\varphi(A,k)$ = if k=0 then null
else assign (delete ($\varphi(A,k)$, i),i,A[i])

where $0 \leq i \leq n$ and $A[i] \neq 0$.

Let the length of a concrete object defined by the formula

$$length_c(c) = k.$$

Then $I_c(c)$: $0 \leq length_c(c) \leq n$.

We next define the length of an abstract object.

$$length_a : structure \rightarrow integer$$

axioms:

$$length_a(null) = 0$$

$$length_a(assing(s,i,e)) = length\ (delete(s,i))+1.$$

Note that one needs to define the properties of the function $length_a$ only for constructor function.

We now define the abstract invariant $I_a(a)$ as follows

$$I_a(a) : 0 \leq length_a(a) \leq n$$

and the abstract bounded objects are defined in terms of

pre- and post-conditions thusly:

{true} s = null {$length_a(s) = 0$}

{length(s') < n} s=assign (s',i,e)

{$length_a(s) = length_a$ (delete (s',i,e))+1}

From the axioms

f(g(x)) = h(x)

the following pre- and post-condition form can be deduced

{y = g(x)} z = f(y) {z = h(x)}

In our example for the abstract selector function read(s,i) we have

{s' = assign (s',i,e)} z = read (s,j)

{<u>if</u> i=j <u>then</u> z=e <u>else</u> z=read (s',j)}

{s' = assign (s",i,e)} s = delete(s',j)

{<u>if</u> i=j <u>then</u> s" <u>else</u> assign (delete (s'',j)i,e)}

We now can use the rule given above, for proving the correctness of an implementation againts an algebraic specification after showing

- that $I_a(s)$ holds for every abstract objects of the given type, and

- that the $length_c$ on concrete objects corresponds to the $length_a$ on abstract objects.

In our example, these can by easily verified.

REFERENCES

[1] Dahl, O.J., Nygaard,K., Myhrhuag,B., The SIMULA 67 common base language, Norwegian Computer Centre, Forskningsveien 1B, Oslo /1968/

[2] Guttag,J.V., Horning,J., The algebraic specifications of abstract data types, Acta Informatica, 10 /1978/ 27-53

[3] Guttag,J.V., Horowitz,E., Musser,D.R., Abstract data types and software validation, Communication of the ACM, 21 /1978/ 1048-1064

[4] Hoare, C.A.R., Proofs of correctness of data representation, Acta Informatica 1 /1972/ 271-281

[5] Lamb S.S. et al. SAMM: A modeling tool for requirements and design specification, Proc. COMPSAC 78, IEEE, /1978/ 48-53

[6] Hamilton,Z., The relationship of design and verification, Journal of Systems and Software, 1 /1979/ 26-32

[7] Lampson,B.W., et all., Report on the programming language Euclid, SIGPLAN Noties 12 /1977/ 2

[8] Liskov, B.H., Zilles, S., Programming with abstract types, SIGPLAN Noties 9 /1974/ 50-59

[9] Liskov, B.H., Zilles,S., Specification techniques for data abstractions, IEEE. Trans. Software Eng. 1. /1975/ 7-19

[10] Teichroew,D., Hershey E.A., "PSLIPSA: A computer-aided technique for structured documentation and analysis of information processing systems, IEEE Trans. Software Eng. 3 /1977/ 41-48

[11] Wulf,W.A., London, R.L., Shaw, M., An introduction to construction and verification of Alphard programs, IEEE Trans. Software Eng. 2 /1976/ 253-265

[12] Zilles, S., Abstract specification for data types, IBM Research Laboratory, San Jose, California /1975/

99/1979 Ivics József: KGST Riga

100/1979 Téli iskola

1980-ban jelentek meg:

101/1980 Gerencsér László - Hangos Katalin:
Diszkrét lineáris sztochasztikus rendszerek
önhangoló szabályozása.

102/1980 Pásztorné Varga Katalin: Rekurziv eljárás

103/1980 Gerencsér Piroska - Szép Endre - Zilahy Ferenc
Marton Zsolt: Robotmegfogók adaptivitása I.

104/1980 Knuth Előd - Radó Péter - Tóth Árpád:
Az SDLA előzetes ismertetése

105/1980 E. Knuth - P. Radó - Á. Tóth:
Preliminary description of SDLA

106/1980 Prékopa András: Sztochasztikus programozási
modellek és alkalmazásuk

107/1980 Kelle Péter: Megbizhatósági készletmodellek
és alkalmazásai

108/1980 Almásy Gedeon: Mérlegegyenletek és mérési
hibák

109/1980 Békéssy A. - Demetrovics J. - Gyepesi Gy.:
Relációs adatbázis logikai szintü vizsgálata
funkcionális függőségek szempontjából

110/1980 Gaál A. - Soltész J. - Ruda M. - Ratkó I.:
Tanulmányok a statisztikai adatfeldolgozásról

111/1980 Benedikt Szvetlána: Nem ismételhető döntéshozatal
analizise kockázattal járó esetekben

112/1980 Verebély Pál: Többprocesszoros, osztott intelligen-
ciáju grafikus rendszerek tervezési és megvalósitási
kérdései